Miguel Henrique Rodrigues de Almeida

Licenciado em Engenharia Informática

Análise de Dados de Atividade de Clientes para Previsão do Nível de Envolvimento

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientadora: Sandra Ingrez, Consultora,
Create IT, Integração e Desenvolvimento de Sistemas Informáticos

Co-orientadores: João Leitão, Prof. Auxiliar,
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

Inês Oliveira, Especialista,
Create IT, Integração e Desenvolvimento de Sistemas Informáticos

Júri

Presidente: Doutora Teresa Isabel Lopes Romão, Profª. Auxiliar, Faculdade de Ciências e Tecnologia da UNL

Arguentes: Doutor Jorg Matthias Knorr, Prof. Auxiliar
Faculdade de Ciências e Tecnologia da UNL
Doutora Inês Oliveira, Especialista, Create IT, Integração e Desenvolvimento de Sistemas Informáticos

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2019

Análise de Dados de Atividade de Clientes para Previsão do Nível de Envolvimento

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Resumo

O tema da aprendizagem automática tem vindo a tornar-se mais popular nos últimos tempos, o que, aliado ao facto de existirem mais e melhores aplicações baseadas na *cloud* leva a que os dois temas se intercetem. Isto leva a que se utilize a aprendizagem automática para evoluir os sistemas que presentemente operam na *cloud*, e em particular nesta tese, uma plataforma que gere o uso de (outros) serviços de *cloud* contratados por um conjunto de clientes.

Com o objetivo de melhorar uma plataforma existente, que permite monitorizar dados de clientes e respetivos dados de utilização de serviços online na *cloud* ao longo do tempo, surge a oportunidade de introduzir modelos de aprendizagem automática para que seja possível passar a catalogar conjuntos de utilizadores com base nas suas preferências. Aliado a isto, existe também a necessidade particularmente desafiante de conseguir prever as necessidades dos diferentes utilizadores da plataforma, prevendo assim eventuais desistências e mudanças de pacote de serviços atuais. Este tipo de previsão permite uma análise muito mais completa dos dados disponíveis sobre os utilizadores e os seus serviços, bem como permite gerar mais valor proveniente da análise dos dados. Neste documento é abordado todo o percurso que culmina na implementação das funcionalidades descritas, desde a criação de um repositório que concilie todas as necessidades da plataforma, passando pelo treino dos modelos com os diferentes conjuntos de dados e finalmente a sua aplicação aos dados reais dos clientes, concluindo assim as previsões e as classificações pretendidas.

Com a aplicação das funcionalidades de classificação de clientes e de previsão de alterações de utilização por parte dos mesmos, passa a ser possível prestar um serviço com maior qualidade e com um grau de personalização completamente diferente do atual, onde o atendimento e o apoio prestado a cada cliente tem um grau de certeza e de individualização bastante superior, aumentando assim a satisfação dos clientes.

Palavras-chave: Aprendizagem Automática, Previsão de Clientes, Classificação de Clientes, Provedor de Soluções na Nuvem, Repositório de dados

Abstract

Machine learning techniques have become increasingly popular in recent times, which, together with the fact that today we have more and more complex applications based on cloud infrastructures have lead these two topics to become entwined, enabling the use of machine learning to evolve systems that are currently operating on the cloud. In particular, in this dissertation, we focus on two platforms that manages the use of (other) cloud services by two sets of clients.

To improve an existing platform, that monitors client data and respective usage of online services by that client over time, the opportunity of introducing machine learning models to make possible the classification of users based on their preferences and usage of services emerges. On top of this, there is also the challenging need to be able to predict the necessities of different users of the platform, as to detect eventual withdrawals and changes to the current service packages contracted. This type of forecast allows a much more complete analysis of the available data regarding the customers and their services, while also allowing to generate value for the company. We aim at the complete implementation of the described functionalities, to create a repository that reconciles all the needs of the platform, then train models for the different datasets, and finally use their application to the actual data of the clients, thus leading to forecasts and the classifications of clients described above.

By applying the customer classification functionalities and predicting changes in their use, it is possible to provide a service with higher quality and with a degree of customization completely different from the one being currently offered by the company, where the service and support provided to each client has a higher degree of certainty and customization, improving overall client satisfaction.

Keywords: Machine Learning, Customer Prediction, Customer Classification, Cloud Solution Provider, Repository of Data

Índice

Lista de Figuras

Lista de Tabelas

Agradecimentos

O trabalho contido nesta dissertação não seria possível sem a ajuda e suporte de várias pessoas. Em primeiro lugar gostaria de agradecer à família, em especial aos pais, por todo o acompanhamento e incentivo durante o decorrer não só da dissertação em si mas também do curso de Engenharia Informática. Aos amigos, por todos os momentos de trabalho mas principalmente de lazer que tornaram este período mais fácil e descontraído.

Gostaria também de agradecer aos orientadores, tanto do lado da faculdade como do lado da empresa pelo tempo despendido e pelo *feedback* dado ao longo do decorrer da dissertação. Deixar um especial obrigado às diferentes equipas da Create IT pelo acolhimento e bom espírito demonstrado, em particular à equipa responsável pelo Cloud Cockpit, pelo tempo despendido em resolução de dúvidas e disponibilidade demonstrada.

Por último, à Dona Ana e ao Senhor Alexandre do K Negra, por todas as noites em que mantiveram o estabelecimento aberto de modo a que festa se fizesse, bem como a todos os que estiveram presentes em tais noites académicas.

Capítulo 1

Introdução

1.1 Contexto do projeto

A palavra *cloud* é uma palavra-chave nos tempos de hoje. Está a tornar-se o principal modelo de negócio para a maioria das indústrias, principalmente no sector tecnológico. Neste sentido surgem os CSP (*Cloud Solution Providers*), que permitem a monitorização do ciclo de vida de um cliente através de soluções de suporte e faturação [30].

Existem dois tipos de CSPs, o indireto e o direto. No que diz respeito ao tipo indireto este aplica-se quando é necessário fornecer serviços aos clientes mas é necessário alguma ajuda ao nível de serviços de infraestrutura. Neste sentido um provedor indireto oferece serviços de faturação, atendimento ao cliente e suporte técnico durante o ciclo de vendas e fecho pós-negócio. Estes serviços controlam inúmeros processos de *backend*, permitindo assim ao revendedor indireto que se foque apenas nas tarefas de encontrar e fechar negócios [30, 31]. Benefícios associados a este tipo de modelos passam pelo apoio fornecido, o portal comercial no qual é possível acompanhar os clientes de forma continua, os guias relacionados com o negócio, que oferecem serviços complementares de forma a aumentar as receitas.

No que toca ao modelo direto, este aplica-se quando o negócio da empresa já opera com infraestruturas de faturação, vendas e suporte funcionais, não necessitando assim que estas sejam adicionalmente fornecidas [31].

A equipa de CSP da empresa Create IT, na qual o trabalho desta dissertação é realizado, desenvolveu a plataforma CloudCockpit, esta plataforma serve de caso de estudo e foco para o desenvolvimento da solução proposta neste documento. O CloudCockpit é um produto de *software* como serviço (do inglês *software as a service*), onde o fornecedor do *software* se responsabiliza por toda a estrutura necessária à disponibilização do sistema e onde o cliente utiliza a plataforma via Internet. Neste caso o CloudCockpit é especializado

em soluções Office 365, Azure, Symantec e neste momento estão em desenvolvimento novas adições.

CloudCockpit é uma plataforma que permite algumas operações, das quais é de destacar as seguintes:

- Permite configurar ficheiros relacionados com a utilização de serviços Microsoft por revendedor e/ou por cliente;
- Mostrar o custo de utilização por subscrição, com a opção de realizar *drill-downs* por categoria e por recurso;
- Apresentar dados de faturação provenientes da Microsoft;
- Alertar o cliente relativamente a renovações de subscrições;
- Atualização automática de listagens de preços baseados na utilização dos serviços e nas licenças adquiridas;
- Mostrar as opções mais vendidas quando são adicionadas novas subscrições, como primeiras ofertas;
- Cálculo de ajustes de preços de subscrições.

A existência de informação sobre os clientes bem como os seus dados de utilização dos diferentes serviços permitem ter uma visão pormenorizada de cada cliente face aos serviços geridos pela plataforma. Neste sentido, é possível controlar os níveis de utilização dos diferentes clientes em diferentes serviços e perceber se os pacotes que estes dispõem estão de facto alinhados com as suas necessidades. Existe também informação sobre todos os pacotes disponíveis, bem como todos os serviços pertencentes aos diferentes pacotes, esta informação é vital para perceber como diferentes pacotes de serviços permitem potencializar as diferentes necessidades dos clientes. Deste modo, a utilização de plataformas como o Cloudcockpit permitem que tanto os revendedores como os clientes tenham acesso aos seus dados de utilização de serviços ao longo do tempo.

1.2 Motivação

Enquanto neste momento já existem plataformas como o Cloudcockpit que permitem aos seus utilizadores, sejam estes *cloud solution providers*, revendedores ou clientes finais, aceder aos seus dados de utilização de diferentes serviços existentes em diferentes pacotes, a evolução do mercado leva à necessidade de que se dê o passo seguinte. Não limitar o acesso à informação daquilo que já existe e aconteceu no tempo, mas sim começar a prever certos acontecimentos e eventos relevantes para o cliente e operação do CSP. Neste sentido, o desafio de melhorar a plataforma Cloudcockpit vem na sequência da nova tecnologia emergente no mercado eletrónico, a aprendizagem automática (do inglês *machine learning*).

O desafio inerente à aplicação de aprendizagem automática na plataforma do Cloudcockpit prende-se com o objetivo de conseguir prever as necessidades dos clientes, de forma a conseguir proporcionar um acompanhamento mais personalizado a cada um. É objetivo deste projeto conseguir utilizar os dados armazenados de forma a treinar modelos de aprendizagem automática que consigam detetar padrões de utilizadores e até mesmo padrões de utilização, de forma a conseguir identificar utilizadores que poderão estar em risco de deixar de ser clientes, anulando assim as suas subscrições aos pacotes disponíveis e permitir uma intervenção antecipada para evitar a perda do cliente.

Á semelhança da previsão de cancelamentos de subscrições é também objetivo deste projeto a previsão detalhada de mudanças de subscrições de pacotes. Esta previsão vem no seguimento de conseguir garantir um apoio mais próximo e personalizado a cada cliente, de forma a que seja possível sugerir os pacotes que melhor se adequam às necessidades de cada cliente em cada situação. Deste modo, é expectável que com a implementação desta funcionalidade seja possível prever quando é que um utilizador necessita de fazer um *upgrade* ou um *downgrade* nas suas subscrições, permitindo assim aos CSP's e aos revendedores sugerir e conciliar essas mudanças de forma atempada.

Outro desafio associado à solução a ser implementada é a possibilidade de agrupar utilizadores de acordo com os seus gastos, necessidades e consumos. Este tipo de agrupamento é bastante comum em sistemas que possuem funcionalidades de *machine learning*, posto isto a utilização dos algoritmos com este fim trará mais possibilidades de implementação de novas funcionalidades na plataforma. Associado à possibilidade de agrupar utilizadores em diferentes grupos está também a possibilidade de deteções de padrões de utilização, podendo assim identificar diferentes clientes que possuam padrões semelhantes e conseguir, de algum modo, utilizar essa informação para melhorar o serviço prestado a esses clientes.

De forma a construir uma solução que contemple todos os desafios mencionados anteriormente é necessário dividir o trabalho em diferentes etapas que irão produzir diferentes contribuições e resultados. Etapas estas enumeradas na Secção 1.3, apresentada abaixo.

1.3 Contribuições previstas

O trabalho a desenvolver consiste na organização dos dados e treino de modelos de aprendizagem automática de forma a conseguir extrair valor acrescentado dos mesmos. De uma forma mais especifica, existem vários pontos a alcançar, nomeadamente:

- Construção de um repositório que concilie os dados dos clientes bem como os diferentes dados de utilização de serviços desses clientes ao longo do tempo;
- Estudo e classificação dos dados disponíveis segundo um grau de relevância para o problema;

- Estudo e análise de diferentes algoritmos de aprendizagem automática a utilizar, no sentido de perceber quais os que garantem um maior grau de sucesso;
- Treino efetivo de modelos de previsão automática aplicados ao domínio aplicacional do problema;
- Análise dos diferentes modelos e algoritmos obtidos;
- Análise dos resultados de previsão face aos objetivos do projeto.

1.4 Organização do documento

O resto deste documento apresenta-se dividido nos seguintes capítulos:

No **Capítulo 2** é fornecido algum contexto sobre a empresa na qual esta tese se enquadra, de forma a perceber que serviços dispõe, bem como quais as necessidades existentes e que visam agora ser combatidas com esta proposta de solução.

O **Capítulo 3** apresenta o estudo relacionado com o tema a ser tratado. São descritos os tipos de aprendizagem automática, sendo esta explicação mais focada na aprendizagem supervisionada, pois esta será o tipo de aprendizagem a utilizar no contexto deste projeto. É também descrito a aplicação da aprendizagem automática quando aplicada à modelação dos utilizadores, bem como os tipos de filtragem de informação que proporcionam um desenvolvimento mais objetivo de soluções.

O **Capítulo 4** apresenta as comparações realizadas entre as diferentes plataforma em estudo, nas quais é possível desenvolver a aplicação que dará solução à proposta desta dissertação.

No **Capítulo 5** é especificada a manipulação feita sobre os conjuntos de dados do problema, bem como o significado e importância de cada métrica presente nos dados. Neste capitulo encontram-se descritos os passos percorridos que culminaram na criação de diferentes conjuntos de dados que serão depois utilizados até ao final da implementação da solução.

No **Capítudo 6** encontra-se a clarificação relativa aos diferentes algoritmos que serão utilizados no decorrer dos testes realizados. É também descrito que métricas de desempenho serão utilizadas para comparar os resultados entre modelos de aprendizagem automática treinados. Por fim consta também informação sobre cada módulo utilizado na plataforma Azure Machine Learning Studio, bem como sobre o fluxo de execução montado na mesma.

No **Capítulo 7** encontra-se descrita toda a informação relativa aos testes realizados sobre os diferentes modelos de aprendizagem automática, desde quais as métricas de avaliação utilizadas, passando pela análise individual de todas as possibilidades de algoritmos a utilizar e culminando com a escolha efetiva da melhor solução ao problema.

O **Capítulo 8** refere o trabalho desenvolvido no que toca à criação de uma aplicação externa que utiliza a solução desenvolvida de forma a classificar grandes quantidades de dados de forma automatizada.

Por fim o **Capítulo 9** faz o fecho desta dissertação, onde são feitas algumas conclusões sobre o trabalho desenvolvido e são ponderados possíveis trabalhos futuros de forma a complementar o que foi desenvolvido até ao momento.

Capítulo 2

Contexto

2.1 História da Create IT

A Create IT foi fundada em 2001 com um princípio orientador: "desenvolver projetos que antecipam o futuro e que representam um efetivo valor acrescentado para o negócio dos nossos clientes."[22]

Sediada na praça de Alvalade em Lisboa, é uma empresa de cariz tecnológico especialista na criação de sistemas multi-plataforma críticos e de suporte ao negócio. Com uma vasta experiência em projetos para clientes de setores tão diversos como as Telecomunicações, Financeiro, Indústria, Distribuição, Turismo e Administração Pública.[22]

2.2 Serviços

As soluções de serviços são focadas no aumento da produtividade de quem as utiliza e são pensadas desde o início para responderem aos desafios do negócio, transformando-o no caminho da otimização e potenciando o seu crescimento.

As soluções colaborativas e adoção do Office 365 são um pilar muito forte na empresa, estas ferramentas permitem que se tire partido de tudo o que o SharePoint e o Office 365 oferecem para partilhar e colaborar com os seus colegas, parceiros e clientes. [24]

Com uma equipa com mais de 15 anos de experiência e como Parceiros Gold da Microsoft na área de Integração de Aplicações, é possível criar soluções empresariais completas para responder às necessidades dos clientes.

2.3 Produtos e Soluções

A Create IT criou internamente um conjunto de produtos que respondem a necessidades do mercado, prontos a utilizar e de acordo com as melhores práticas do mercado.[22]

O SmartPortals é a plataforma de eleição para o rápido desenvolvimento de portais, já que a sua extensibilidade suporta a integração com os sistemas dos clientes. Tendo por base o CMS Umbraco, gerência de conteúdos assente em tecnologia *opensource*, permite gerir conteúdos de forma ágil e responde a todas as dimensões de informação, destacando-se pelas suas características de escalabilidade.[41]

O DiggSpace é um portal de comunicação interna pronto a usar que assenta em SmartPortals, tirando partido das funcionalidades do Office 365. Com o DiggSpace a empresa promove a produtividade, o conhecimento e a eficiência, permitindo reforçar o sentimento de pertença, colaboração, envolvimento, compromisso e partilha entre todos os colaboradores.[41]

O CloudCockpit é uma consola web para gerir de forma centralizada a rede CSP de produtos *cloud*, permitindo gerir a informação do cliente e controlar as subscrições. Como está disponível num modelo de software como serviço (do inglês Software-as-a-Service (SaaS)), não há qualquer investimento inicial nem custos de instalação e configuração para clientes.[41]

2.4 Relevância do projeto

O tópico desta tese insere-se no produto CloudCockpit da empresa, este permite controlar e gerir informações dos clientes e controlar as diferentes subscrições que estes possuem. Atualmente a plataforma permite recolher dados de utilização de diferentes serviços *cloud*. Neste sentido é possível acompanhar o crescimento de um cliente ao longo do tempo no que diz respeito ao seu nível de utilização de serviços Microsoft.

Existem diferentes tipos de serviços, começando pelos serviços como o Azure, os quais funcionam com base na *cloud* e são faturados de acordo com a utilização que o utilizador faz dos serviços. O tipo de dados que o CloudCockpit permite acompanhar relativamente a estes serviços de utilização passam desde os dados básicos de faturação, como as datas em que um serviço é faturado, a região de faturação, bem como os preços para os diferentes revendedores ou clientes do serviço e os impostos inerentes às taxas cobradas pelos serviços. Existe também informação especifica sobre o serviço que está a ser faturado, desde o seu tipo, como por exemplo, se é um serviço de armazenamento, de máquinas-virtuais e se estes possuem replicação, cópias de segurança ou outras definições associadas. Todos estes dados estão relacionados com o cliente final, o provedor do serviço e o revendedor do mesmo, sendo que todas estas informações são também disponibilizadas na plataforma.

Para além dos serviços que são faturados com base na utilização existem ferramentas que são faturadas com base em pacotes de subscrições. É o caso de serviços do Office 365

como o Skype para empresas, o SharePoint, o Teams e muitos outros.

No serviço por subscrição o tipo de dados são um pouco diferentes quando comparados com os dados dos serviços faturados por utilização. Existe a informação relativa ao nome do serviço e a que tipo de pacote este pertence, e existe a noção de licenças adquiridas e licenças utilizadas, isto porque um cliente pode adquirir várias subscrições e não as ativar para utilização. No entanto, analogamente aos serviços faturados por utilização, existe também a informação financeira associada às diferentes subscrições, como os custos associados à licença, ao revendedor e ao cliente final.

O trabalho proveniente da tese em questão tem como objetivo melhorar a plataforma atual, utilizando os diferentes dados de forma a conseguir construir modelos de aprendizagem automática que consigam prever situações que necessitem de uma potencial intervenção. Estas intervenções podem passar por sugerir a um cliente um *upgrade* ou *downgrade* do pacote de serviço, de forma a ir ao encontro do que melhor serve o cliente. Outro aspeto importante é conseguir prever se um cliente está em risco de deixar de o ser, alertando assim para uma possível intervenção atempada de forma a evitar que se perca um cliente. Estas novas possibilidades de prever o que melhor se adequa a um cliente com base nos seus dados, vem aumentar o valor do CloudCockpit, melhorando assim o serviço que é prestado.

2.5 Sumário

Neste Capítulo foi dado algum contexto sobre o âmbito no qual se desenvolve esta dissertação, de forma a que se perceba que tipo de empresa e que tipo de produto está em causa. Relativamente à relevância do projeto foi dado algum enquadramento sobre o tipo de interação que existem entre ferramentas bem como qual o objetivo da presente tese face àquilo que existe neste momento na empresa.

No Capítulo que se segue (Capítulo 3) serão abordadas algumas temáticas relativas a trabalhos relacionados com esta dissertação, passando por diferentes tipos de aprendizagem automática bem como os seus desafios, é também abordada a temática dos sistemas de filtragem de informação, vendo diferentes abordagens realizadas por outros autores em diferentes problemas similares.

Capítulo 3

Trabalho Relacionado

Neste capítulo é apresentado e discutido trabalho relevante para a solução a desenvolver, contendo assim uma síntese inicial de trabalho relacionado. Em particular são debatidos os seguinte tópicos:

Na Secção 3.1 são abordadas diferentes formas que possibilitam a aprendizagem automática.

Na Secção 3.2 as especificações deste tipo de aprendizagem automática bem como os seus problemas e técnicas associadas são discutidas.

Na Secção 3.3 apresenta a aplicação direta de técnicas de aprendizagem automática associada às ações e padrões dos utilizadores.

Por fim, na Secção 3.4 aborda os sistemas de filtragem de informação e a sua mais valia no tratamento de dados.

3.1 Tipos de aprendizagem automática

As pessoas podem cometer erros durante a análise, ou possivelmente, quando tentam estabelecer relações entre diferentes métricas, isto torna difícil encontrar soluções para determinados problemas. A utilização de aprendizagem automática pode muitas vezes ajudar na solução destes problemas, aumentando a eficiência dos sistemas.

Existem diferentes tipos de aprendizagem automática [25], pelo que a forma como a informação é apresentada aos algoritmos de aprendizagem têm influência na forma como estes processam a mesma e tiram conclusões úteis à resolução dos problemas.

3.1.1 Aprendizagem por implantação

Quando estamos na presença de uma aprendizagem por implantação direta de novo conhecimento não há inferência ou outras transformações de conhecimento no lado do

algoritmo de aprendizagem. Variantes deste tipo de aquisição de conhecimento incluem:

- Aprender por ser programado, ou seja, o mecanismo de aprendizagem é construído ou modificado por uma entidade externa, o que não apresenta nenhum tipo de esforço do lado do algoritmo no que toca à inferência de padrões e conclusões, visto que este é definido explicitamente pelo programador.[25]
- Aprender por memorização de factos sem inferências sobre a informação. O termo *rote learning* é utilizado primariamente neste contexto, sendo que a aprendizagem se dá por repetição. Este é o método que mais se aproxima da aprendizagem realizada pelo ser humano, uma vez que a informação é aprendidas através da sua captação por repetição.[25]

3.1.2 Aprendizagem por instruções

Numa transposição para o mundo real, é possível mapear este tipo de aprendizagem como aprender através de um professor ou outra fonte organizada de informação. Requer que o aprendiz transforme o conhecimento oferecido de uma certa linguagem de input para uma representação interna utilizável, e utilize essa mesma informação de forma a integrar esta com a informação retida previamente de forma útil. O aprendiz tem de realizar algum trabalho, através de inferência, mas a maior fração do trabalho realizado continua do lado do professor, que apresenta e organiza o conhecimento de uma forma que aumenta incrementalmente o conhecimento do aluno.

3.1.3 Aprendizagem por analogia

Adquirir novos artefactos de conhecimento transformando e aumentando conhecimento existente que é similar àquele que é desejado no novo conceito da nova situação. Um sistema que aprende por analogia pode ser aplicado para converter um programa de computador existente e funcional, num que realiza funções similares para o qual não foi originalmente desenhado. Aprender por analogia requer mais inferência da parte do algoritmo do que a aprendizagem por implantação.

3.1.4 Aprendizagem por exemplos

Dado um conjunto de exemplos e contra-exemplos de um conceito, o algoritmo induz um conceito geral que descreve os exemplos positivos e nenhum dos contra-exemplos. É uma abordagem fortemente investigada na inteligência artificial. A quantidade de inferência feita pelo algoritmo é muito maior do que na técnica de aprendizagem por instruções, visto que não existem exemplos gerais a serem dados por uma entidade exterior, e é também uma inferência de alguma forma maior do que na técnica de aprendizagem por analogia, pois não existem conceitos gerais a serem dados como "sementes" dos quais novos conceitos podem surgir. Aprendizagem por exemplos pode ser sub-categorizada dependendo do tipo de fonte dos dados, nomeadamente:

- Se a fonte for externa e segura, que conhece o conceito e gera sequências de exemplos que têm como objetivo ser o mais úteis possíveis ao processo de aprendizagem.
- A fonte ser o próprio algoritmo, normalmente este conhece o seu estado atual de conhecimento, mas não tem informação sobre o conceito objetivo a ser adquirido. Assim sendo o algoritmo pode gerar instâncias na informação básica que acredita serem necessárias para descriminar as diferentes descrições de contexto.
- A fonte ser um ambiente externo, neste caso o exemplo é gerado aleatoriamente, em que o algoritmo deve confiar em observações relativamente não controladas.
- Apenas exemplos positivos, em que todos os exemplos providenciados são positivos. Estes têm um efeito negativo que é o facto de não providenciarem informação para prevenir generalizações excessivas. Neste tipo de aprendizagem, o excesso de generalização deve ser evitado de forma a considerar apenas a menor quantidade possível de generalizações, ou através de conhecimento fornecido a priori para inferir conceitos futuros.
- Serem fornecidos exemplos positivos e negativos. Neste tipo de situação os exemplos positivos forçam o aparecimento de generalizações, contudo a existência de exemplos negativos previnem o excesso de generalizações. Esta é a forma típica de aprendizagem por exemplos, sendo que a maioria dos sistemas de aprendizagem automática utiliza técnicas desta classe.

Aprendizagem por exemplo pode ser realizada de uma só vez ou de forma incremental. A abordagem incremental aproxima-se da aprendizagem humana, que permite que o algoritmo utilize conceitos parcialmente aprendidos e permite à fonte de informação focar-se nos aspetos básicos de um novo conceito antes de se focar em aspetos menos centrais. Por outro lado, a aprendizagem realizada de uma só vez é menos apta a guiar para caminhos negativos, pois os exemplos iniciais que são fornecidos apresentam a totalidade dos casos positivos e negativos, pelo que o algoritmo consegue logo à partida fixar a sua base de conhecimento.[25]

3.1.5 Aprendizagem por observação e descoberta

Também chamado de *unsupervised learning*, este é uma forma bastante geral de induzir aprendizagem que inclui sistemas de descoberta, tarefas de formação de teoria e criação de critérios de classificação. Este tipo de aprendizagem obriga o algoritmo a realizar mais inferência do que qualquer tipo de abordagem discutida até agora. O algoritmo não tem qualquer tipo de conjunto de instância com conceitos particulares, nem é dado acesso a um elemento exterior que consegue classificar instância internamente geradas como instância positivas ou negativas para um dado conceito.

No entanto, no caso da técnica de aprendizagem por observação e descoberta existem dois tipos de observações possíveis de serem realizadas:

- **Observação passiva:** Onde o observador classifica múltiplos aspetos sobre o ambiente, sem alterar os mesmos.

- **Observação ativa:** Onde o observador perturba o ambiente para observar os resultados das suas perturbações. Á medida que um sistema vai adquirido conhecimento isto leva à confirmação ou negação das suas teorias, o que torna a exploração diferente à medida das necessidades e da evolução de execução do algoritmo.

3.2 Aprendizagem Automática Supervisionada

Cada instância de um conjunto de dados utilizado por algoritmos de aprendizagem automática é representado por um conjunto de atributos. Aos diferentes atributos podemos também atribuir o nome de métricas. As métricas podem ser continuas, categóricas ou binárias. Se as instâncias apresentarem etiquetas (*labels*), que correspondem aos corretos outputs ou classes às quais pertencem as instâncias, então estamos diante da utilização da técnica de aprendizagem automática supervisionada.

O objetivo deste tipo de aprendizagem automática passa por construir uma base de conhecimento através da análise dos dados em função das suas etiquetas. Desta forma, treinando os modelos de aprendizagem automática com um conjunto de dados de treino, é possível que este perceba de que modo os diferentes valores de cada métrica influencia na obtenção da respetiva etiqueta que classifica aquela instância. O que possibilita que o modelo consiga classificar um novo conjunto de dados prevendo as etiquetas de saída desse novo conjunto. [6]

A construção de um modelo de aprendizagem automática supervisionada necessita de algum cuidado no que toca à forma como o modelo retira conclusões sobre a informação do conjunto de dados. Existem por isso três categorias para classificar o tipo de modelo de aprendizagem automática obtido.[27]

- Pouco adaptado *(Underfitted)*, onde o modelo de aprendizagem não conseguiu retirar informação útil proveniente do conjunto de treino, desta forma, a atribuição da etiqueta às novas instâncias de dados não apresenta os melhores resultados. Isto acontece pois existiu uma falha no que toca à aprendizagem sobre as relações no conjunto de treino.

- Demasiado adaptado *(Overfitted)*, na qual o modelo de aprendizagem automática fez uma análise demasiado especifica ao conjunto de treino, resultando assim num modelo que depende em demasia dos dados de treino, o que faz com que a previsão de novas instâncias não seja a melhor, pelo simples facto destas não possuírem os exatos valores dos dados do conjunto de treino.

- Adaptação necessária *(Good Fit)*, quando foi possível evitar situações tanto de *overfitting* como de *underfitting*. Isto torna o modelo de aprendizagem automática mais

robusto, adequando-se o necessário a novos dados mas conseguindo prever corretamente as respetivas etiquetas.

Figura 3.1: Representação visual dos diferentes tipos de adaptação aos dados (Adaptado de [4])

3.2.1 Dificuldades gerais da aprendizagem supervisionada

O primeiro passo é reunir um conjunto de dados e identificar quais as métricas que são mais relevantes. Como é possível que o conjunto de dados possua algum ruído no que toca à representação dos dados, bem como casos onde alguns valores são omissos, é necessário realizar um pré-processamento.

Tipos de dados que podem estar errados:

- Valores impossíveis (*impossible values*) ou duvidosos que tenham sido introduzidos manualmente ou incorretamente obtidos.
- Não ter sido introduzido qualquer valor (*missing value*).
- Valores irrelevantes estarem a ser apresentadas.

Se o ruído nos dados, por exemplo (*impossible values*), não poder ser resolvido de outra forma, como poder editar os dados de forma a que estes sejam reintroduzidos, então estes devem ser categorizados como *missing values* e deve-se simplesmente eliminar a informação do conjunto de dados de entrada.

Informação incompleta é um problema inevitável quando lidamos com informação real. Geralmente existem fatores importantes no que toca ao processamento de valores desconhecidos. Um dos mais importantes é a fonte dos valores desconhecidos:

- O valor é desconhecido porque foi esquecido ou perdido;
- Uma certa métrica não é aplicável numa dada instância;
- Para uma dada observação, o valor daquela métrica não importa para o treino do modelo.

Dependendo da circunstância, existem diferentes tipos de métodos nos quais podemos escolher para tratar os *missing values*, melhorando assim a qualidade geral do conjunto de dados de entrada.

A seleção de um subconjunto de métricas é o processo de identificar e remover a maior quantidade de métricas redundantes e irrelevantes possível. Isto reduz a dimensionalidade do nosso conjunto de dados e permite que os algoritmos de *data mining* operem com maior rapidez e eficiência [23]. O facto de algumas métricas dependerem de outras induz problemas no que toca à exatidão de modelos de classificação supervisionada. Este problema pode ser abordado através da construção de novas métricas a partir do conjunto de métricas base.

A avaliação do classificador é usualmente obtida através da exatidão das previsões. Existem pelo menos três formas de classificar a exatidão do classificador. A primeira é dividir o conjunto de treino, utilizado dois terços para treino do modelo e um terço para estimar a performance. Outra técnica, conhecida como validação cruzada, é dividir o conjunto de treino em subconjuntos com a mesma dimensão e mutuamente exclusivos, sendo que para cada subconjunto o classificador é treinado na união de todos os outros subconjuntos. *Leave-one-out validation* é um caso especial da validação cruzada. Todos os subconjuntos são testados, este tipo de validação é mais dispendioso em temos de computação, mas útil quando é requerido que se tenha uma estimação da taxa de erro mais precisa [59].

3.2.2 Árvores de Decisão

As árvores decisão são árvores que classificam instâncias através da sua ordenação consoante os valores das suas métricas. Cada nó numa árvore de decisão representa uma métrica numa instância a ser classificada, sendo que cada vértice representa o valor que o nó (métrica alvo) pode assumir.

A métrica que melhor divide o conjunto de treino deve ser o nó raiz da árvore. Existem muitas maneiras de descobrir a métrica que melhor divide a árvore, e diversos estudos [25] acabaram por concluir que não existe um único "melhor método". O procedimento é depois repetido em cada partição do conjunto de dados, criando sub árvores até que o conjunto de treino esteja dividido em subconjuntos da mesma classe.

Existe dois tipos de abordagem que um algoritmo baseado numa árvore de decisão pode utilizar de forma a evitar *overfitting* dos dados durante o treino:

- Parar o algoritmo de treino antes que este faça *fit* ao conjunto de treino.

- Escolher entre diferentes árvores. Se duas árvores empregam o mesmo tipo de testes e têm a mesma exatidão de previsão, então normalmente a que tiver menos folhas é considerada preferencial para o treino.

3.2.3 Conjuntos de regras de aprendizagem

Árvores de decisão podem ser transformadas em conjuntos de regras através da criação de regras separadas para cada caminho da raiz da árvore até cada folha.[38] Contudo, regras também podem ser induzidas diretamente a partir do conjunto de treino, utilizando uma variedade de técnicas, como foi estudado por Quinlan. [38]

O objetivo é criar o menor conjunto de regras que é consistente com o modelo de dados de treino. Uma grande quantidade de regras aprendidas normalmente é indicador de que um algoritmo de aprendizagem com algum grau de *overfitting*.

A característica mais útil associada aos conjuntos de regras de aprendizagem prende-se com a sua compreensibilidade. Para a tarefa de aprendizagem de problemas binários, regras são mais compreensíveis do que árvores de decisão, pois normalmente as abordagens relacionadas com regras aprendem um conjunto de regras apenas para as classes positivas. Por outro lado, a definição para objetos de múltiplas classes continua por aprender, o algoritmo de regras deve ser executado separadamente para cada uma das classes separadamente no caso da classificação de múltiplas classes.

3.2.4 Redes neuronais

Outro tipo de algoritmo bem conhecido no que toca à aprendizagem automática são os algoritmos que têm por base a noção de perceptrão.

Um perceptrão é uma camada simples de uma rede neuronal, sendo que que ao conjunto de múltiplas camadas de perceptrões se dá o nome de rede neuronal. Um perceptrão é um classificado linear (binário), que funciona de forma bastante simples: [48]

- Os valores de chegada são multiplicados por um peso W, definido nas propriedades do perceptrão.
- Os valores multiplicados são depois agregados num valor denominado de soma ponderada.
- Esta soma ponderada é depois aplicada a uma função de ativação. O exemplo mais simples de uma função de ativação é o caso da função *unit step*, na qual é retornado o valor 1 no caso do valor da soma ponderada ser positivo e 0 no caso da soma ponderada ter um valor negativo.

A forma mais comum de uso destes algoritmos é a aprendizagem através de um conjunto de instâncias de treino correndo o algoritmo repetidamente através do conjunto de treino até que este encontre um vetor de predição que é correto para todo o conjunto de treino.

Peceptrões só conseguem classificar instâncias que sejam linearmente separáveis. Se uma linha reta ou plano poder ser desenhada para separar instâncias provenientes do input nas suas categorias corretas, então estas instâncias são linearmente separáveis e o

peceptrão poderá encontrar uma solução para o problema. Caso isto não aconteça então nunca haverá o ponto em que todas as instâncias são classificadas corretamente. As redes neuronais artificiais foram criadas com o intuito de resolver este problema.

Uma rede neuronal consiste num grande número de peceptrões juntos através de um padrão de conexões. Existem três tipos de classes, as unidades de input, que recebem a informação a ser processada, as unidades de output, onde os resultados do processamento são encontrados e as unidades escondidas que permanecem no meio destas. Geralmente a determinação do tamanho apropriado da camada escondida é um problema, porque uma estimativa baixa do número necessário pode levar a uma má aproximação e à generalização das capacidades, enquanto que um excesso de nós pode resultar em *overfitting* e eventualmente fazer a procura pela solução ótima mais difícil [25].

Uma rede neuronal depende de três coisas, do input, das funções de ativação e do peso atribuído a cada conexão de input. Sendo que os primeiros dois aspetos são fixos resta apenas controlar o comportamento da rede neuronal através da alteração dos pesos atribuídos.

Apesar de as redes neuronais e as árvores de decisão serem fundamentalmente diferentes, é possível fazer a algumas considerações gerais:

- As redes neuronais normalmente conseguem providenciar mais conhecimento incremental do que as árvores de decisão.
- O tempo de treino das redes neuronais é normalmente superior ao tempo de treino das árvores.

3.2.5 Máquinas de Suporte Vetorial

Máquinas de Suporte Vetorial (SVMs) são uma técnica de aprendizagem automática supervisionada recentemente introduzida. As SVMs funcionam em redor da noção de margem, o lado de um hiperplano que separa duas classes de informação, maximizando a margem e as instâncias de cada lado deste.

Para os casos em que temos informação que seja linearmente separável, assim que o hiperplano seja encontrado os pontos que ficam na margem deste são conhecimentos como vetores de suporte e a solução é representada como uma combinação linear destes pontos. Sendo que outros pontos são ignorados. Por isto a complexidade de um modelo SVM não é afetado pelo número de métricas que o conjunto de treino possui [35]. Sendo este um aspeto bastante importante a considerar no momento da escolha do algoritmo a utilizar na construção de modelos de aprendizagem automática.

Do ponto de vista de eficiência computacional não é necessário passar a informação por um processo de seleção de métricas, enquanto que com técnicas de aprendizagem como as árvores de decisão ou redes neuronais começa a ser algo complicado obter bons resultados se passamos a treinar conjuntos de dados com algumas centenas de métricas [29].

Se a SVM não conseguir achar a solução devido ao facto de o conjunto de entrada ter instâncias mal classificadas, o problema pode ser abordado através da suavização das margens, aceitando assim alguns erros de classificação para as instâncias de treino. No mundo real existe muita informação que não é linearmente separável, para isto não existe um hiperplano que consiga separar as instâncias positivas das negativas no conjunto de treino. Uma solução para esta limitação é o aumento da dimensionalidade e definição de um hiperplano separado.

No entanto os métodos SVM são binários, pelo que no caso de problemas com múltiplas classes deve-se reduzir o problema para um conjunto de múltiplos problemas de classificação binária, permitindo assim a classificação de várias classes distintas.

3.2.6 Comparação de técnicas de aprendizagem

Olhando para os diferentes atributos que compõem o conjunto de dados, estes podem ser denominados como métricas. As diferentes métricas são então analisadas e selecionadas de acordo com a possivel contribuição que estas podem oferecer na solução do problema.

Geralmente máquinas de suporte vetorial e redes neuronais tendem a ter um melhor desempenho no que toca a problemas com múltiplas dimensões e métricas continuas. Em contraste, sistemas baseados em lógica tendem a ter um melhor desempenho quando se lida com métricas discretas ou categóricas.

No que toca à interpretação, algoritmos baseados em lógica são fáceis de interpretar, sendo que as redes neuronais e SVMs são notórios pela sua baixa interpretabilidade, este tipo de fatores pode ser preponderante na escolha do algoritmo a utilizar.

Não existe um algoritmo de aprendizagem que tenha um melhor desempenho que todos os outros para todos os conjuntos de dados, pelo que para resolver a questão "Que algoritmos usar para ter um resultado com maior exatidão?" a melhor abordagem é estimar a exatidão dos algoritmos candidatos para o problema e selecionar aquele que apresente melhores resultados. Esta avaliação é algo que terá de ser efetuado para cada conjunto de dados e que está dependente do objetivo do problema. Pelo que diferentes problemas terão diferentes conclusões no que toca ao algoritmo que permite melhores resultados.

O conceito de combinar classificadores é proposto como uma nova maneira de melhorar o desempenho dos classificadores individualmente. O objetivo dos algoritmos de classificação é gerar resultados com maior precisão e exatidão [28]. Sendo que é útil utilizar os pontos fortes de um algoritmo para complementar os pontos fracos de outro, melhorando assim a solução final.

3.3 Aprendizagem Automática aplicada à modelação de utilizadores

Quando falamos de modelação dos utilizadores e das suas ações, estas podem ser tão variadas quanto os diferentes propósitos para os quais os modelos de utilizadores são

formados. É possível estar na presença de diferentes tipos de análise, nomeadamente:

- Os processos cognitivos que levaram os utilizadores a realizar certas ações;
- As diferenças entre as capacidades do utilizador e as capacidades de um *expert*;
- Padrões comportamentais de um utilizador ou as suas preferências;
- Características do utilizador.

Estes modelos podem conter informação pessoal relativa ao utilizador e informação que é depois adicionada ao utilizador que pode ou não estar diretamente ligada à adaptação do sistema a esse utilizador, mas que pode ser utilizada para categorizar o utilizador num de vários estereótipos, o que torna possível ao sistema antecipar alguns dos comportamentos desse utilizador [42].

Existem até sistemas educacionais que utilizam técnicas de modelação para personalizar o processo de aprendizagem, de forma a tornar o processo mais adaptativo às capacidades dos alunos, tendo em conta as suas habilidades e o seu contexto histórico, bem como prever ações dos estudantes. Isto permite proporcionar um ensino mais personalizado [21, 55].

Métodos de aprendizagem automática têm sido aplicados a problemas de modelos de utilizadores principalmente para construir modelos de utilizadores individuais aque interagem com sistemas de informação [3, 5, 39, 40].

Uma dimensão bastante importante é a necessidade de distinguir as abordagens que dizem respeito a se o modelo é direcionado para os utilizadores individuais ou para comunidades (grupos) de utilizadores [54].

Embora investigação académica relacionada com aprendizagem automática se concentre mairoritariamente na modelação individual dos utilizadores [3, 5, 39, 40, 54], estão a emergir aplicações baseadas em aprendizagem automática no que toca ao comercio eletrónico que relacionam a formação de modelos genéricos de comunidades de utilizadores.

3.3.1 Personalização e recomendação de informação

O objetivo da personalização de informação é poder proporcional ao utilizador aquilo que este quer ou necessita sem que este tenha de pedir explicitamente. [32]

Neste tipo de sistema existe a distinção entre as informações dos utilizadores e dos objetos.

- **Informação relativa aos objetos:** Inclui conteúdo descritivo sobre os objetos ou produtos que estão em estudo para serem recomendados.
- **Informação relativa aos utilizadores:** Inclui referências passadas dos dados de utilização de serviços ou de *ratings* passados atribuídos pelo utilizador, bem como dados pessoais ou demográficos.

Existem também diferentes tipos de personalização de informação relativamente ao tipo de *feedback* que utilizam por parte dos utilizadores, nomeadamente abordagens reativas e pro-ativas. Abordagens reativas utilizam processos convencionais que requerem que o utilizador realize interações explicitas, quer na forma de questão (do inglês *queries*) quer na forma de *feedback* incorporado nos processo de recomendação [2]. Em sistemas de *feedback* baseados em *ratings*, os utilizadores têm de classificar todas as recomendações que são feitas, de acordo com o quanto estas se adequam àquilo que estes procuram. No que toca aos sistemas que utilizam *feedback* de preferência, ao utilizador é apresentada uma lista com recomendações e este é motivado a escolher apenas um produto, sendo este aquele que mais se adequa aos seus requisitos [8, 19].

Outro tipo de abordagem é a abordagem pro-ativa, que aprende sobre as preferências do utilizador e recomenda objetos com base na informação aprendida, não necessitando que o utilizador introduza o seu *feedback* de forma explicita [2].

Um exemplo deste tipo de abordagem é o que acontece em plataformas como a Netflix, onde os hábitos de consumo dos utilizadores são utilizados para gerar novas sugestões de séries e filmes, de acordo com as suas preferências, sendo que estas sugestões serão depois apresentadas ao mesmo.

3.3.2 Desafios

O treino de modelos de aprendizagem automática tem alguns desafios, identificados e discutidos em baixo:

Sistemas como o de Syskill e Webert [36], que era aplicado ao problema de recomendação de sites web, em que os utilizadores à medida que navegam na Internet dão o seu *feedback* sobre as páginas web visitadas, carregando num botão de aprovação ou desaprovação, apresentam uma limitação inerente a este tipo de aplicações de aprendizagem automática, é impossível construir um modelo com uma exatidão aceitável a não ser que se tenha um número relativamente grande de exemplos para ter como base. É por isso normal que um algoritmo de aprendizagem necessite de diferentes exemplos de treino [52].

Outro desafio direto da aplicação de aprendizagem automática a muitas tarefas de modelos de utilizadores é que as abordagens através de aprendizagem supervisionada necessitam que exista uma etiqueta (*label*), ou um rótulo explicito para a informação recebida, contudo este rótulo pode não ser atribuído corretamente através da observação do comportamento do utilizador.

Por exemplo, no sistema de Syskill e Webert era necessário que o utilizador realizasse tarefas extra de forma a que fosse possível ter estas etiquetas preenchidas, contudo isto pode ser um problema, pois a maioria dos utilizadores não realiza este tipo de tarefas extra se não existir uma necessidade de realizar as mesmas ou se não existir um incentivo associado à sua realização.

De forma a endereçar este problema, uma das possíveis soluções é inferir as *labels*

através do comportamento dos utilizadores, verificando se estes realizam um dado conjunto de ações, e conforme as ações realizadas é atribuída uma das *labels*. Outra forma de resolver o problema passa por utilizar um pequeno conjunto de dados já catalogados com as respetivas *labels* para assim conseguir inferir a *label* correta num conjunto maior de dados, que serão depois utilizados para treinar os algoritmos de aprendizagem.

A modelação de utilizadores é algo que é muito suscetível a mudanças ao longo do tempo, uma vez que os atributos que caracterizam os utilizadores têm tendência a variarem ao longo do tempo. Assim sendo é necessário que os algoritmos de aprendizagem sejam capazes de se ajustar a estas mudanças com alguma rapidez. Este fenómeno, do ponto de vista da aprendizagem automática, tem o nome de *concept drift*. [56]

A ideia central de ajuste ao *concept drift* passa pela utilização de uma janela ajustável, onde o tamanho da janela depende dos indicadores observados, como certas mudanças em temos de distribuição. Algumas soluções, como a de Chiu e Webb [10], estudaram a indução de um duplo modelo de utilizadores, sendo que é fácil concluir que informação mais recente sobre os utilizadores espelha melhor o atual conhecimento, preferências ou estado atual de um utilizador do que informação de tempos passados. Contudo esta informação mais recente pode levar a modelos demasiado específicos. De forma a contornar este problema, Chiu e Webb utilizaram este modelo com dupla informação, em que primeiramente os algoritmos de aprendizagem utilizam as informações mais recentes para o seu treino, contudo, no caso da previsão não ter sido obtida com suficiente confiança são então consultados os dados mais antigos [10] de forma a evitar que modelo formado seja demasiado específico.

O crescimento da Internet tem tido um tremendo impacto no que toca à aprendizagem automática aplicada à modelação de utilizadores. Se por um lado a Internet tem levado a um novo caminho com novas oportunidades para auxiliar os utilizadores com diferentes serviços, através da utilização dos detalhes de utilização destes, por outro lado o aumento da informação disponível, bem como o aumento do número de utilizadores online criou novos desafios relacionados com a complexidade computacional. Investigação realizada num ambiente académico mostra que existe uma preocupação relativa à complexidade computacional[54]. Quando um novo algoritmo é proposto, é comum que sejam feitas diferentes avaliações da forma como este se comporta em diferentes situações de processamento, considerando sempre a sua taxa de exatidão. Por esta razão não é errado que um algoritmo que apresente uma exatidão de 78% seja escolhido na vez de um algoritmo que garanta 80% de exatidão, pois provavelmente o tempo de execução deste segundo é consideravelmente mais elevado e como tal, impede um melhor funcionamento e processamento dos dados de um cenário com dados do mundo real e em que a latência do processamento é relevante.

3.4 Sistemas de filtragem de informação

De forma a que os modelos de aprendizagem automática possam sugerir aos utilizadores certos objectos e fazer a filtragem da informação, é necessário utilizar diferentes métodos de filtragem consoante o objetivo do problema. Podemos por isso dividir o tipo de método a utilizar em dois tipos. O primeiro tipo é uma filtragem baseada no conteúdo (*content-based*), na qual o sistema aceita a informação que descreve os vários objetos, e aprende a prever quais os objetos que se enquadram no modelo de utilizador. Este tipo de filtragem só pode ser aplicada quando existem objectos cujas propriedades e valores de atributos sejam descritos. O outro tipo é a filtragem colaborativa ou social (*collaborative-based)*, na qual o sistema actualiza o modelo de utilizador e prevê os objectos que se adequam a este, baseado no *feedback* obtido por parte dos diferentes utilizadores. Numa análise colaborativa um objeto é considerado interessante para um utilizador se, para outros utilizadores com gostos semelhantes, aquele objeto também era de interesse. A preferência com base na análise colaborativa tem sido utilizada em sistemas especialistas em filtragem e retorno de informação [11].

Contudo a abordagem *content-based* e *collaborative-based* não se excluem uma à outra, de facto estas podem ser combinadas de forma a criar aquilo que se chama um modelo de utilizador de preferências integradas. Em particular este tipo de modelo é construído nos termos de um conjunto de atributos predefinidos, tal como no caso dos modelos de preferência dos utilizadores *content-based*, contudo existem dois atributos nesta abordagem integrada que se caracterizam por uma abordagem *collaborative-based*, na qual um dos atributos caracteriza o utilizador e o outro caracteriza um objeto[11].

De forma a que seja possível construir automaticamente um modelo de utilizador de preferências integradas, um método de aprendizagem indutiva é aplicado a um conjunto de entradas do conjunto de dados [3].

Existe uma grande quantidade de informação que é criada e enviada através de meios de comunicação eletrónica. Estas grandes quantidades de informação revelam-se difíceis de gerar valor se não forem selecionadas, tratadas e filtradas de acordo com um objetivo específico. Desta forma, existem processos de filtragem de informação que permitem utilizar a informação proveniente dos diferentes serviços de forma a que seja gerado valor acrescentado a esta. Na figura 3.2 consta uma representação das diferentes fases de um sistema de filtragem de informação, normalmente composto por quatro componentes básicos: (a) um componente de análise de informação, (b) um componente de filtragem, (c) um componente de modelo de utilizador e (d) um componente de aprendizagem.

- O componente de análise da informação (a) obtém e guarda objetos de informação junto dos provedores dos dados. A informação é analisada e apresentada no formato apropriado para o problema. Esta representação será o input do componente de filtragem (b).

- O componente do modelo de utilizador (c) quer implicitamente e/ou explicitamente

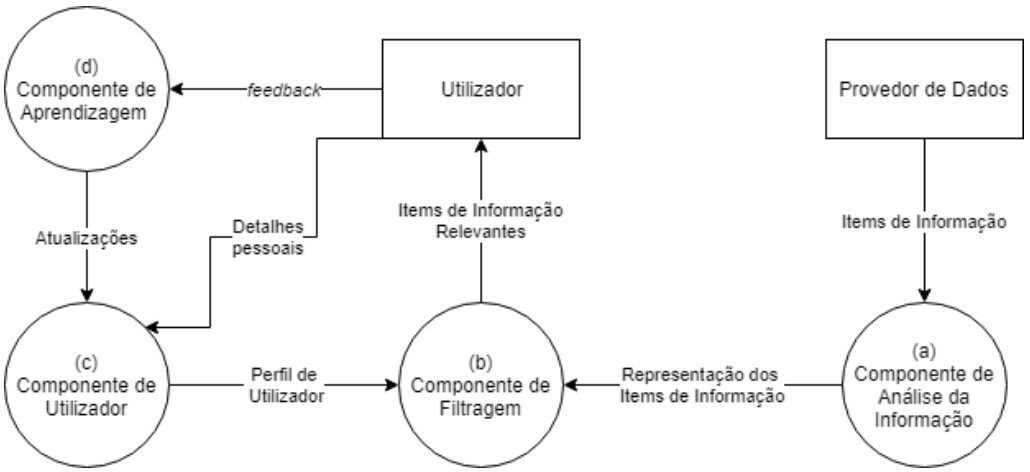


Figura 3.2: Modelo genérico de filtragem de informação

recolhe informação sobre os utilizadores e as suas necessidades e constrói um modelo de utilizadores. O modelo de utilizador será também input para o componente de filtragem.

- O componente de filtragem (b), que é o coração do sistema de filtragem de informação, faz a correspondência entre os perfis dos utilizadores e os objetos de informação representados e decide se um dado objeto na coleção é relevante para o utilizador.
- O componente de aprendizagem (d) é necessário para que se melhore a filtragem. Devido à dificuldade em modelar os utilizadores e devido também às alterações nas suas informações, os sistemas de filtragem necessitam que os processos de aprendizagem detetem estas mudanças nos interesses dos utilizadores. Caso contrário existiriam incoerências que iriam afetar os resultados da filtragem, deturpando os resultados finais.

3.5 Sumário

Neste capitulo foram clarificados alguns conceitos base que serão utilizados no resto do documento para explicar o trabalho realizado. Passando pelos diferentes tipos de aprendizagem automática, bem como a elucidação do tratamento que a informação do conjunto de dados deve receber antes que esta possa ser utilizada para treino de modelos de aprendizagem automática. De forma mais profunda foi debatido o tema da aprendizagem automática de tipo supervisionado, uma vez que este será o tipo de aprendizagem utilizado para dar resposta às necessidades da empresa Create IT.

Por último, ainda relativo ao trabalho relacionado, é também demonstrado como a aprendizagem automática pode ser utilizada no contexto da modelação de utilizadores, de forma a filtrar, compreender e assimilar características dos diferentes utilizadores, com o objetivo de melhorar um serviço ou produto, aumentando o nível de personalização do mesmo.

No próximo capitulo encontra-se descrita a ponderação e escolha da plataforma a utilizar para implementação da solução ao problema, onde são comparadas duas plataforma distintas de fabricantes diferentes.

Seguidamente é também descrita a manipulação efetuada sobre o conjunto de dados original, de forma a tornar estes dados utilizáveis no contexto da plataforma escolhida e do problema.

Capítulo 4

Plataforma a Utilizar

É necessário que exista uma exploração que possibilite a escolha da melhor plataforma a utilizar para o treino dos modelos de aprendizagem automática. Uma vez que a empresa para qual a solução em questão se destina é parceira da Microsoft e da Amazon. Existem assim duas plataformas *cloud* que podem ser utilizadas no desenvolvimento desta etapa. Neste sentido foi também objetivo do trabalho realizar um estudo pormenorizado tanto sobre a plataforma Azure Machine Learning Studio[1] como a plataforma Amazon Web Services, Machine Learning.[2]

No sentido de perceber qual das duas plataformas poderá originar melhores resultados, é necessário que se efetue um estudo e documentação dos diferentes aspetos que cada plataforma proporciona, bem como o seu grau de fiabilidade e estado atual de desenvolvimento e suporte das ferramentas disponíveis em ambas as plataformas. Após perceber quais os pontos fortes e limitações de cada uma das plataformas é necessário escolher, justificadamente, uma das duas plataformas de forma a que seja possível começar o treino de modelos de aprendizagem automática que garantam os melhores resultados possíveis. Sendo que é objectivo destas plataformas conseguir realizar de forma rápida uma aprendizagem supervisionada sobre o conjunto de dados fornecido, de forma a detetar padrões que ajudem tanto na previsão de perdas de clientes como na previsão de alterações de pacotes.

4.1 Custo de utilização

Os custos associados à utilização do serviço são um fator importante no que toca à decisão de qual das duas plataformas utilizar, não só pelos custos associados a uma primeira fase

[1] https://azure.microsoft.com/pt-pt/services/machine-learning-studio/
[2] https://aws.amazon.com/pt/machine-learning/

de estudo e conceção do sistema, mas também porque a longo prazo estes custos são suportados ou pela própria empresa ou pelos seus clientes, o que afeta o preço/lucro final do serviço que está a ser disponibilizado, tornando-se um ponto preponderante no mercado.

No que toca à plataforma da Amazon esta apresenta uma desvantagem relativa ao carregamento dos dados, uma vez que estes têm de ser guardados em AWS antes de poderem ser utilizados nos serviços de Machine Learning [34], o que vem acrescentar à equação os custos de armazenamento dos dados, enquanto que do lado do Azure estes podem ser importados diretamente dos servidores SQL da empresa já montados e operacionais.

Do lado da Amazon Machine Learning há dois custos a considerar: taxas de computação e taxas mensais de previsão. O preço de computação é de 0,37€ por hora e as taxas mensais de previsão têm um custo de 0,089€ por cada 1000 previsões. [43]

Por outro lado, da parte da Azure Machine Learning Studio existe o custo de 0,844€ por hora de computação e um custo fixo de 8,425€ por mês por *workspace* existente.[45]

De forma a termos uma noção de como estes preços influenciam a fatura final, tomemos o exemplo de utilizar 20 horas de tempo de computação para gerar os modelos, obtendo um total de 890,000 previsões. Do lado da Amazon as 20 horas de computação somariam um total de 7,40€, taxas de previsão formam um total de 79,21€ ((0,844€/1000)*890,000), desta forma a soma destes valores totaliza o valor de 86,61€ nesse mês exemplo, fora custos extra de armazenamento. Do lado da Azure, as 20 horas de computação totalizam 16,88€, sendo que a este valor temos de acrescentar os custos fixos de termos uma workspace Azure Machine Learning (8,425€), fazendo assim o total de 25,305€ nesse mês.

Com estes resultados tornou-se claro que a diferença de preços entre as duas plataformas leva a um favorecimento da escolha do Azure Machine Learning Studio, contudo vale a pena ainda assim estudar os serviços providenciados por cada plataforma, pois embora mais barato, o Azure poderá não disponibilizar todos os serviços necessários à resolução do problema.

4.2 Serviços de Análise Preditiva Disponibilizados

Embora ambas as plataformas proporcionem muitas opções e diferentes métodos de abordar diferentes problemas, como pode ser consultado na documentação oficial de cada plataforma [44, 46], existe um âmbito que é fulcral no trabalho a ser desenvolvido, nomeadamente a análise preditiva. Isto porque o problema a solucionar prende-se com a deteção de casos em que existem clientes em risco de saída bem como na previsão de situações em que é útil a um cliente realizar um *upgrade* ou *downgrade* à sua subscrição.

As duas plataformas continuam em constante desenvolvimento, pelo que é possível que com o passar do tempo possuam cada vez mais opções e abranjam mais e diferentes métodos. Contudo, como é possível ver na Tabela 4.1, retirada de [1], existem algumas diferenças naquilo que são as opções disponibilizadas nas duas plataformas até à data.

	AWS ML	Azure ML
Classificação	**Sim**	**Sim**
Regressão	**Sim**	**Sim**
Clustering	**Sim**	**Sim**
Recomendação	Não	**Sim**
Deteção de Anomalias	Não	**Sim**
Ranking	Não	**Sim**

Tabela 4.1: Serviços disponibilizados pelas duas plataformas em análise [1]

No que toca a realizar a classificação de se um cliente está ou não em risco de deixar de ser cliente ambas as plataformas poderiam dar resposta ao problema, uma vez que ambas permitem a utilização de módulos de classificação. De um ponto de vista de extensibilidade do trabalho, pode fazer sentido requerer que a plataforma tenha opções para realizar regressões, por exemplo de custos associados a serviços, bem como pode fazer sentido realizar operações de *clustering*, de forma a agrupar conjuntos de clientes com base nos seus atributos, e no que toca a todos estes pontos ambas as plataformas satisfazem a totalidade dos requisitos. As diferenças surgem quando falamos no ponto de recomendar aos clientes *upgrades* ou *downgrades* aos seus pacotes de serviços. No caso da AWS Machine Learning este tipo de recomendação ainda não é suportada, o que causa um entravo claro à resolução do problema da sugestão de pacotes de serviços.

O facto do Azure Machine Learning permitir não só processar recomendações de pacotes mas também tendo o extra de permitir a detenção de anomalias nos dados vem contribuir com mais um ponto a favor do Azure. Isto mostra que neste momento a ferramenta da Azure é mais completa e apresenta melhores condições para dar resposta ao problema.

4.3 Decisão

Apesar de ambas as plataformas possuírem algumas condições vantajosas para a implementação dos trabalhos, a plataforma da Azure revelou-se preferível para seguir com a ordem de trabalhos. Em primeiro lugar porque apresenta um total de custos associados bastante diminuto quando comparada com a plataforma fornecida pela Amazon. Aliado a isto o facto da ferramenta da AWS não suportar recomendação de pacotes torna-se uma grande limitação ao projeto.

De acordo com os pontos referidos anteriormente podemos concluir que a melhor plataforma para dar seguimento aos trabalhos é sem dúvida a plataforma Azure Machine Learning Studio.

4.4 Sumário

Neste Capítulo foi abordada a escolha da plataforma que será a base da implementação de toda a solução. A escolha da plataforma teve em conta dois pontos preponderantes, o custo associado à utilização de cada uma das plataformas e o tipo de funcionalidades que são disponibilizadas nas mesmas. Com esta análise foi possível determinar que a plataforma que reúne melhores condições é de facto a plataforma de Azure Machine Learning Studio.

O próximo Capítulo (5) é dedicado à manipulação do conjunto de dados, de forma a criar agregações dos dados que serão depois utilizadas no decorrer da solução, desta forma, o Capítulo encontra-se dividido em diferentes secções, partindo desde a filtragem e tratamento dos dados até à criação das diferentes agregações.

Capítulo 5

Manipulação do Conjunto de Dados

De forma a possibilitar o desenho, execução e validação dos resultados para os diferentes tipos de modelos de aprendizagem automática é necessário que exista uma implementação que possa ser executada em tempo real e a partir de qualquer computador, como tal selecionamos a plataforma Microsoft Azure Machine Learning Studio de forma a tornar isto possível.

Ao longo deste Capítulo iremos abordar os diferentes procedimentos efetuados nas diferentes fases de desenvolvimento, desde a filtragem e tratamento dos dados do problema (Secção 5.1), passando pela criação de diferentes agregações de dados a utilizar (Secção 5.2, combinando as diferentes agregações com diferentes possibilidades de algoritmos para o treino dos modelos (Secção 6.1), de forma a garantir os melhores resultados possíveis. Após o treino efetivo dos modelos de aprendizagem automática com base nas escolhas feitas (Secção 7.3) é abordada a temática da criação de um *web service* (Capítulo 8) que possibilite a utilização do serviço por fontes externas.

5.1 Filtragem e Tratamento dos Dados

De forma a ser possível realizar a filtragem e tratamento dos dados do conjunto de dados inicial foi necessário, numa primeira fase, realizar o reconhecimento daquilo que eram os dados e o seu significado. No contexto do domínio abordado nesta dissertação existem dois tipo de dados, os dados de subscrições e os dados de utilização.

Dados de Subscrições No que toca aos dados de subscrições, estes dizem respeito a dados de clientes relativos às suas subscrições de serviços *cloud*, incluindo o seu custo, nomeadamente produtos de Office 365 como é o caso de produtos como o Yammer, SharePoint, Exchange, Teams, entre outros. As métricas que formam o conjunto de dados inicial

relacionado com subscrições a produtos encontram-se descritas no Apêndice A, onde é indicado o tipo de métrica e respetiva descrição. Os pacotes de subscrições caracterizam-se pelo facto dos clientes adquirirem subscrições de um certo produto e pagarem por número de ativações do serviço, não dependendo assim do tipo de utilização do produto que fazem mas sim se estão ativas ou não, assim sendo, se um cliente tiver uma subscrição ativa este será faturado da mesma forma quer utilize ou não o serviço.

Dados de Utilização A principal diferente deste tipo de dados, relativamente aos dados de subscrições recai no facto da faturação feita aos clientes depender diretamente do uso que estes dão aos serviços, por exemplo, um cliente que possui uma quantidade superior de gigas de informação armazenados irá pagar uma taxa de armazenamento maior. Os dados de utilização fazem-se representar por algumas métricas diferentes, descritas no Apêndice B.

Numa primeira fase filtraram-se os dados iniciais retirando as colunas correspondentes a identificadores internos dos dados, nomeadamente os ID's. No caso dos dados de subscrições as colunas descartadas foram o Id, ResellerId e ProviderId, no que toca aos dados de utilização juntam-se a estas as colunas de ParterId, PartnerBillableAccountId, MpnId, Tier2MpnId, ExternalOfferId, entre outras colunas responsáveis por identificadores. De forma a ser mais fácil visualisar as alterações ao conjunto de dados, a Tabela 5.1 apresenta as diferentes métricas que foram retiradas a cada conjunto de dados. O propósito por detrás da exclusão destas colunas recai no facto dos identificadores não possuírem nenhuma informação concreta nem sobre o tipo de serviço nem sobre que tipo de utilização que o mesmo tem, assim sendo, a sua presença no momento em que os algoritmos matemáticos tentam estabelecer relações entre os dados, de forma a gerar modelos de aprendizagem automática, apenas iria causar ruído e diminuir a taxa de sucesso dos mesmos.

Como os dados em questão são dados reais, é normal que existam algumas entradas que apresentem dados não definidos, ou seja, campos vazios. A existência de campos com valor indefinidos deve ser corrigida, com o objetivo de prevenir eventuais problemas que possam surgir quando estamos a treinar um modelo de aprendizagem automática. [47]

A abordagem ao tratamento de campos vazios fez-se recorrendo ao módulo disponibilizado pelo Azure Machine Learning Studio denominado como Apply SQL Transformation, efetuou-se por isso o tratamento dos dados através de queries SQL, editando assim os dados de entrada e permitindo que para todos os casos em que os dados estivessem não definidos se pudesse atuar em concordância. No caso das colunas numéricas, optou-se pela afetação do valor zero, enquanto que para colunas como CustomerPriceMarginRule e SubscriptionPriceMarginRule foi definida a utilização de um novo valor categórico '*None*'. Desta forma os dados foram transformados de forma a que não existisse nenhum campo com entradas não definidas, possibilitando assim o avanço na ordem de trabalhos.

Do ponto de vista de *feature engineering* foi necessário alterar a estrutura dos dados. Os dados relativos à análise de subscrições contêm dois tipo de subscrições, as subscrições

Tipo de Dados	
Subscrições	**Utilização**
Id	Id
ResellerId	ResellerId
ProviderId	ProviderId
SubscriptionID	PartnerId
	MpnId
	Tier2MpnId
	ExternalOfferId
	ResourceGuid
	PriceListId
	PartnerBillableAccoutnId
	ResellerIndirectPartnerId
	CustomerId
	CustomerCompanyName
	OrderId

Tabela 5.1: Colunas retiradas aos dados para filtragem e tratamento

de licenças Microsoft e as subscrições de Azure, apesar destes segundos serem dados de utilização estão englobados no conjunto de subscrições devido ao facto de pertencerem a pacotes maiores que proporcionam ao cliente ter subscrições de produtos e usufruir de serviços online disponibilizados pela plataforma Azure. A existência destes dois tipos de faturação presentes no mesmo conjunto de dados leva à necessidade de haver colunas diferentes para representar algo similar, como é o caso das colunas: TotalLicenseCost, TotalUsageCost, DailyLicenseCost, DailyUsageCost, TotalLicenseCostForReseller, TotalUsageCostForReseller, TotalLicenseCostForCustomer, TotalUsageCostForCustomer, DailyLicenseCostForReseller, DailyUsageCostForReseller, DailyLicenseCostForCustomer, DailyUsageCostForCustomer. Quando as colunas que dizem respeito a uma subscrição do tipo *License* estão preenchidas, todas as colunas que dizem respeito a uma subscrição de *Usage* encontram-se vazias e vice-versa. De forma a diminuir o número total de colunas e ajustar os dados de uma forma mais cómoda, foi necessário modificar as métricas do conjunto de dados.

Foram criadas novas métricas, nomeadamente TotalCost, DailyCost, SubscriptionType, TotalCostForReseller, TotalCostForCustomer, DailyCostForReseller e DailyCostForCustomer. O intuito destas cinco métricas é substituir completamente as doze métricas mencionadas anteriormente. A métrica SubscriptionType é uma métrica categórica, que permite distinguir se estamos na presença de uma entrada que diz respeito a uma subscrição de *License* ou de *Usage*, enquanto que as restantes métricas fabricadas têm a função de apresentar a informação relativa aos custos diários e totais da subscrição, independentemente do tipo desta.

Do lado dos dados de utilização também existiu a necessidade de criação de novas

métricas, uma vez que os dados iniciais possuíam a informação relativa à ConsumedQuantity e ConsumedQuantityMonth, verificou-se que muitas vezes a quantidade consumida num dia era superior à quantidade consumida para o mês inteiro até à data, desta forma foi criada a métrica ConsumedQuantityRate, que resulta da razão ConsumedQuantity/-ConsumedQuantityMonth. Com esta métrica conseguimos perceber as flutuações que existem a nível de consumos para os vários clientes, percebendo assim se no panorama geral que é o consumo de um cliente no mês, aquele dia em especifico teve um consumo particularmente elevado ou baixo.

Foram ainda introduzidas três novas métricas ao conjunto de dados, estas métricas têm como propósito aumentar a informação relativa à quantidade de serviços que não estão a ser utilizados e que continuam a ser pagos, portanto a quantidade de investimento que é desperdiçado por cliente. Estas métricas foram criadas de forma a perceber se a quantidade de falta utilização de um serviço influencia a permanência dos clientes ou as trocas de subscrições de produtos. Sendo assim foram criadas a métricas de nome PercentQualActive, UnusedCost e CostPerSeat, onde PercentQualActive é obtida através da razão entre as licenças ativas e as licenças compradas, permitindo assim perceber a percentagem de licenças que estão ou não a ser utilizadas. Relativamente à métrica de UnusedCost esta é obtida através de uma fórmula que nos permite determinar o total monetário que não está a ser utilizado pelo cliente, face àquilo que é pago no total e poderia ser utilizado. Por fim, a métrica CostPerSeat permite saber qual o custo associado a um colaborador para um certo pacote, isto porque para cada subscrição existe um preço e um número máximo de colaboradores que podem usufruir do produto, o CostPerSeat é obtido tendo em conta o preço da subscrição e o número de colaboradores que está efetivamente a utilizar o produto.

Com as alterações efetuadas ao conjunto de dados é possível então começar a construir diferentes agregações dos dados, a criação destas advém da necessidade de ter os melhores resultados possíveis, assim sendo, é objetivo deste trabalho perceber que tipo de agregação de dados proporciona um melhor resultado final para os diferentes algoritmos matemáticos a utilizar. Segue-se por isso (Secção 5.2) a descrição do trabalho efetuado com o objetivo de selecionar a melhor agregação de dados e o melhor algoritmo a aplicar para o treino de modelos de aprendizagem automática.

5.2 Seleção do Conjunto de Dados

É proveitoso para o estudo e aprendizagem de como os diferentes conjuntos de dados influenciam o resultado final e que se perceba qual a melhor especificidade dos dados. Com isto em mente, foram realizados vários testes na qual se foi alterando a granularidade dos dados, bem como foram sendo testados diferentes algoritmos de aprendizagem automática para cada uma das diferentes granularidades, permitindo assim um estudo exaustivo de todos os algoritmos em função de todas as agregações de dados criadas, o

que conduziu à escolha de quais as combinações de dados/algoritmos a utilizar para dar solução ao problema.

De forma a perceber o que foi feito relativamente ao conjunto de dados inicial existem dois conceitos a compreender: granularidade e agregação.

Relativamente à granularidade dados dados esta define-se pelo detalhe presente nos dados. Quando temos uma menor granularidade dos dados, ou granularidade fina, isto significa que temos mais detalhe nos dados, ou seja, é possível consultar todas as pequenas informações e detalhes dos dados. À medida que a granularidade vai aumentando (granularidade grossa) vamos tendo cada vez menos detalhes na informação, pois começam-se a abstrair certos tipos de dados e conceitos. A noção de granularidade e de detalhe nos dados é inversamente proporcional. [53]

Quanto ao termo de agregação este é um tipo de processo onde a informação contida nos dados é pesquisada, filtrada e apresentada de forma sumariada de acordo com os objectivos do negócio. A agregação de dados pode ser feita de forma manual ou com a utilização de *software* especializado.[51]

Dado que ao longo de toda a implementação da solução o trabalho foi dividido em duas partes, faz sentido que também exista esta divisão no que toca à documentação do trabalho, pelo que todos os processos e alterações feitas aos dados irão ser apresentados em duas partes, a primeira relativa aos dados de subscrições (Secção 5.2.1) e segunda relativa aos dados de utilização (Secção 5.2.2).

5.2.1 Dados de Subscrições

Começando pelos dados pertencentes às subscrições dos clientes foi possível elaborar um total de seis agregações diferentes, formando assim um total de seis hipóteses de conjuntos de dados para utilização nos diferentes testes.

5.2.1.1 Criação de diferentes conjuntos de dados

A primeira agregação a ser formada foi uma agregação que diz respeito apenas aos clientes, na qual foi priorizada a coluna do CustomerId em função da data. Esta agregação permite-nos assim abstrair o conceito de subscrição ou serviço em particular e olhar apenas para o cliente como um total ao longo do tempo, assim sendo, um cliente que possua diversas subscrições vai ter estas subscrições agregadas num só valor final, possibilitando assim analisar a evolução do cliente como um todo, em função do tempo. Esta agregação caracteriza-se por possuir uma entrada por cliente para cada ponto no tempo em que se realizou uma observação dos dados de entrada.

A segunda agregação feita tem em atenção as diferentes subscrições dos vários clientes, dando assim valor à coluna do CustomerId e SubscriptionId nas várias datas. Esta agregação define-se pelo aumento do detalhe dos dados relativamente à primeira agregação feita, passando agora a diferenciar todas as subscrições de todos os clientes. Com esta segunda agregação é possível perceber como as diferentes subscrições individuais

de cada cliente evoluem ao longo do tempo, passa a ser possível, não só ter informação relativa aos diferentes clientes mas também passa a ser possível ter informações relativas à sazonalidade de serviços e tendências de evolução de serviços. Caracteriza-se assim pela existência de uma entrada para cada subscrição de cada cliente sempre que se realizou uma extração de dados.

Relativamente às últimas quatro agregações de informação estas podem ser divididas em dois grupos, as primeiras duas agregações utilizam a coluna Date para fazer a agregação, sendo que esta coluna representa a data efetiva na qual foi feita a leitura/extração dos dados, enquanto que para as restantes duas agregações foi tida em conta a coluna ProcessedDateTime, que utiliza a data na qual os dados são processados para faturação. Mais uma vez o intuito de testar os agregados utilizando diferentes intervalos de tempo vem no seguimento de perceber se uma agregação temporal maior tem benefício quando comparada com uma agregação temporal menor, isto porque a coluna Date tem muito mais ocorrências do que a coluna ProcessedDateTime, e isto fica a dever-se ao facto da coluna Date ter extrações quase diariamente enquanto que a coluna ProcessedDateTime apresenta apenas extrações semanais ou até mesmo quinzenais no caso de alguns produtos.

A terceira e quarta agregação regem-se pela coluna Date, sendo que para a terceira é utilizada a noção de cliente e produto, representados pelas colunas CustomerId e Product respetivamente. Os diferentes produtos em questão dizem respeito a pacotes de produtos, como é o caso do Office 365 Business Premium, Office 365 Business Essentials e muitos outros pacotes de serviços Microsoft. Assim sendo, é possível utilizar esta agregação para ver a evolução dos clientes em função do tempo relativamente à sua utilização de todo o pacote de serviços, uma vez que o produto engloba dentro de si diferentes serviços disponibilizados à subscrição. O que distingue a quarta agregação da terceira é a possibilidade de observar mais detalhadamente os mesmos dados, agregando assim os dados por cliente, produto e serviço, representados pelas colunas CustomerId, Product e Workload respetivamente. Ao acrescentar a especificidade do serviço passa a ser possível analisar os dados com um elevado grau de minúcia, pois é possível ver os consumos de serviços específicos dentro dos diferentes pacotes, como é o caso de serviços como o Yammer, Exchange, Sharepoint e muitos outros. Este tipo de visão sobre os dados pode ser útil para conseguir sugerir pacotes de serviços que vão ao encontro das necessidades e do uso que os clientes fazem dos serviços.

Por fim, relativamente à quinta e sexta agregação realizada sobre os dados de subscrições estas têm por base a mesma estrutura da terceira e quarta, diferem destas pois utilizam as entradas de tempo presentes na coluna ProcessedDateTime para agrupar os dados em vez de fazerem uso da coluna Date.

5.2.2 Dados de Utilização

No que toca à criação das diferentes agregações com os dados de utilização de serviços por parte dos clientes, foram criadas duas agregações distintas. Um ponto que difere o tipo de agregações feitas com os dados de utilização daquelas que foram feitas com os dados de subscrições é o facto destas necessitarem de informações provenientes da tabela de subscrições, isto pois nos dados originais relativos aos dados de utilização apenas constam as informações relativa às subscrições, sendo que não existe uma correspondência de quem é o cliente a quem esta pertence. De forma a corrigir esta limitação é então utilizada a tabela subscrições na qual é possível fazer a relação entre o identificador da subscrição e o cliente que a adquiriu.

5.2.2.1 Criação de diferentes conjuntos de dados

Uma das agregações feitas tem como objetivo utilizar diferentes rácios como métricas de aprendizagem, assim sendo é uma agregação que aliada aos dados informativos da subscrição e do produto especifico junta informação presente nas colunas ConsumedQuantity, ConsumedQuantityMonth e ConsumedQuantityRate. Esta informação permite-nos perceber respetivamente, a quantidade de serviço consumida/utilizada naquela data especifica, o total acumulado naquele mês até então e o rácio entre as duas.

A segunda agregação feita com este tipo de dados pretende perceber se em vez de fazer uso dos rácios de utilização é mais proveitoso utilizar as taxas associados ao serviço, como tal, aliada à informação da subscrição e do produto especifico juntou-se a informação relativa aos custos antes e após a aplicação de taxas de impostos, preços relativos para o cliente, para o revendedor e taxas associadas à própria subscrição, entre outros dados numéricos associados a custos e taxas do serviço.

Com a criação destas duas agregações pretende-se fazer a distinção de duas vertentes. A vertente relativa a todas as taxas e variáveis que afetam o custo final do serviço daquela que usa os valores associados ao próprio consumo/utilização do serviço por parte do cliente.

5.3 Sumário

Uma vez criadas todas as agregações de dados pretendidas é possível montar um ambiente de testes de forma a conseguir testar todas as agregações de dados para os diferentes algoritmos e medir o seu desempenho face às alterações. De forma a que seja possivel compreender de que forma serão testadas as diferentes possibilidades, o Capítulo 6 apresenta as especificações sobre os algoritmos matemáticos, métricas de avaliação e ambiente de testes utilizado nos testes.

Capítulo 6

Algoritmos, Métricas e Ambiente de testes

Antes de se abordar o treino e teste dos vários modelos de aprendizagem automática é necessário discutir os diferentes algoritmos que estão por detrás dos modelos, bem como quais serão as métricas utilizadas para avaliar a prestação dos diferentes modelos de forma a poder escolher qual a melhor combinação de algoritmo/agregação a utilizar.

O tipo de aprendizagem automática utilizado neste problema define-se como aprendizagem supervisionada, ou seja, os diferentes algoritmos aprendem através de dados etiquetados. Depois de estabelecer uma correlação entre os dados, os algoritmos determinam que etiqueta mais se adequa a uma certa entrada nos dados, baseando-se em padrões e associações estabelecidas no momento do treino dos algoritmos [49].

No caso concreto do problema de previsão de abandono de clientes (*churn*) ou de previsão de situações de *upgrade* e *downgrade* de pacotes, a tarefa em questão visa prever qual a categoria de uma certa entidade nos dados de input (neste caso um cliente), a partir de um modelo de aprendizagem automática baseado nas variáveis fornecidas, quer numéricas quer categóricas. O output do classificador é portanto uma categoria do problema, correspondente à observação feita pelo classificador.

6.1 Algoritmos em Análise

A documentação do Azure Machine Learning Studio disponibiliza não só a total documentação de todos os algoritmos disponíveis para a criação de modelos de aprendizagem automática mas também providencia alguma ajuda na seleção de que algoritmo utilizar em função do problema a resolver [58]. Assim sendo, por se tratar de um problema de classificação entre duas classes (*churn* ou não) fez-se uso dos algoritmos de classificação de duas classes (*Two-class*). Algoritmos disponibilizados na plataforma como é caso do algoritmos de máquinas de suporte vetorial (*support vector machines*) normalmente

adequam-se a problemas com muitas variáveis, nomeadamente mais de cem, assim sendo foi possível descartar os algoritmos Two-class SVM e Two-class locally deep SVM do conjunto de hipóteses, pois o problema em questão não possui esse número de variáveis. Como o preço final da implementação da solução é uma prioridade relevante, o tempo de execução do algoritmo de classificação é crucial para a escolha, sendo que isto leva à exclusão do algoritmo de redes neuronais (Two-class neural network), pois a execução deste tipo de algoritmos é normalmente mais demorada que a dos restantes algoritmos, acrescendo assim em custos de execução, custos estes que em ultima análise seriam ou suportados pela própria empresa ou pelo cliente final, aumentando assim o preço atribuído ao serviço. Restam por isso testar os diferentes resultados da execução dos diferentes conjuntos de treino para os algoritmos que se seguem:

6.1.1 Two-class averaged perceptron

Um perceptrão no que diz respeito ao tema da aprendizagem automática é utilizado para classificadores binários, podendo assim classificar entre duas classes. É um tipo de classificação linear, ou seja, é um algoritmo que faz as previsões baseadas numa função de previsão combinada com a atribuição de pesos (W) a um conjunto de variáveis (X) [57].

Na Figura 6.1 encontra-se representado o processo de aceitação das variáveis e respetivos pesos, para que no fim seja produzido um único output com a classificação.

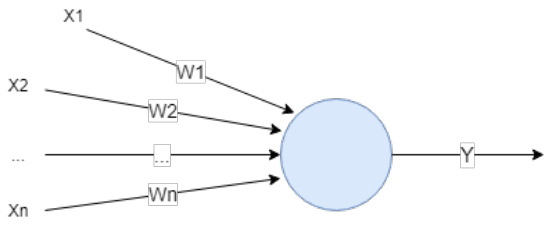


Figura 6.1: Representação de um perceptrão

O uso de perceptrões na classificação de entidades permite efetuar a separação linear de classes, ou seja, permitir achar um vetor cujo hiperplano separa as entidades positivas das negativas, sendo que estas devem ser linearmente separáveis, ou seja, deve ser possível traçar uma divisão entre as diferentes classes, divisão esta que pode ter uma margem maior ou menor, dependendo do erro que estamos dispostos a tolerar [9]. Demonstrado na Figura 6.2.

Assim sendo, procura-se a menor margem possível, onde para um vetor w e um conjunto de treino S, a margem de W é dada por:

$$\gamma = min\, x \epsilon S \, \frac{|< w, x >|}{||x||}$$

Que representa o mínimo ângulo do cosseno entre w e x, onde x pertence a S. Permitindo assim perceber qual a menor margem que permite a divisão de classes das entidades em análise [57]. Podendo esta ser grande ou pequena, como é visível na Figura 6.3 abaixo:

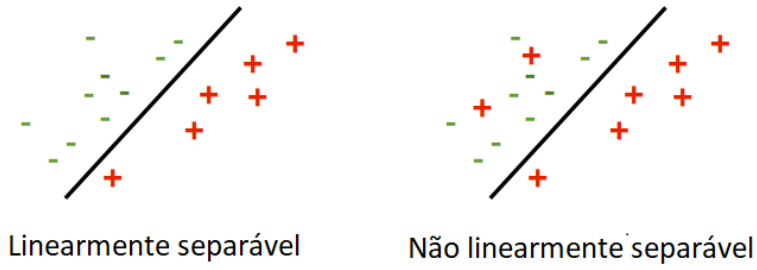


Figura 6.2: Separação Linear

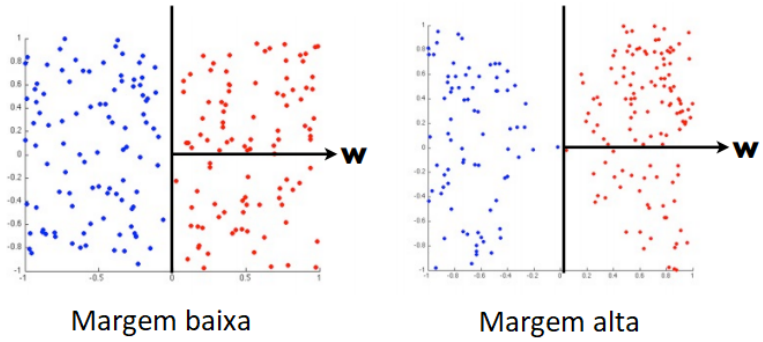


Figura 6.3: Diferença entre margens divisórias com um baixo e alto valor

Assim sendo, para classificar uma entidade, dado o conjunto de variáveis que são recebidas como input do perceptrão, bem como o vetor de pesos ponderados que são calculados de forma a conseguir dividir linearmente as duas classes em estudo, o perceptrão atribui o valor de 0 ou 1 ao output. É este output que é utilizado para classificar uma entidade do conjunto de dados, pois esta estará de um dos lados da linha que divide as duas classes.

Uma das vantagens associada ao uso do algoritmo de *averaged perceptron* é o facto dos perceptrões apresentarem um bom desempenho no que toca a previsões com base em períodos grandes de tempo [26], permitindo assim que os dados dos clientes ao longo do tempo sejam utilizados da melhor forma.

6.1.2 Two-class logistic regression

Uma regressão logística é similar à regressão linear mas é utilizada quando a variável dependente não é um número mas sim uma classe, neste caso o sim e o não. A regressão logística permite-nos estimar a probabilidade associada à ocorrência de algo acontecer face a um conjunto de variáveis [50].

O modelo da regressão linear é dado pela relação entre variáveis (X) combinadas linearmente com pesos ou valores de coeficientes (representados pela letra grega β) de forma a obter o valor y, formando assim a seguinte equação:

$$y = \beta 0 + \beta 1X1 + ... + \beta nXn$$

Como a variável a ser modelada pode apenas assumir dois valores (0 e 1), então a probabilidade do input (X) pertencer a uma classe (Y=1) pode ser escrito formalmente da seguinte forma:

$$P(X) = P(Y = 1|X)$$

[7]

Uma regressão logística é um método linear, mas as previsões são transformadas utilizando a função logística.[7] Isto faz com que deixemos de percecionar as previsões como combinações lineares dos inputs como é feito nas regressões lineares, o que faz com que modelo possa ser escrito da seguinte forma:

$$P(X) = \frac{e^{\beta 0+\beta 1*X}}{1 + e^{\beta 0+\beta 1*X}}$$

Por fim a probabilidade logarítmica de ocorrer uma classe e a outra não ocorrer classifica a entrada com base na maior probabilidade de uma das classes. Adicionando o logaritmo natural (ln) à equação, esta pode ser expressa da seguinte forma:

$$ln(\frac{p}{1-p}) = \beta 0 + \beta 1x$$

Tendo assim um output linear à direita e um input à esquerda que é o logaritmo da probabilidade de uma certa classe. A classificação de uma entrada dá-se consoante o resultado de probabilidade obtido, se este for menor que 0,5 então a entrada pertence à classe 0, de não existir uma situação de perda de cliente, caso seja igual ou superior a 0,5 então pertence à classe 1 que representa a existência de perda de cliente [49].

6.1.3 Two-class Bayes point machine

O módulo disponibilizado pelo azure machine learning studio utiliza uma abordagem Bayesiana para classificação linear.

A regra de Bayes baseia-se na seguinte regra:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Onde no contexto da classificação é possível substituir Y pela classe em estudo e X por um conjunto de métricas que definem uma entidade, ou seja, as diferentes variáveis. Como X é um conjunto, podemos substituir o X na equação expandindo a mesma obtendo:

$$P(Y|X1, ..., Xn) = \frac{P(X1|Y)P(X2|Y)...P(Xn|Y)P(Y)}{P(X1)P(X2)...P(Xn)}$$

Com esta equação é possível obter os valores olhando para o conjunto de dados e substituir os mesmos na equação, o que faz com que o denominador não se altere para

todo o conjunto de dados, sendo assim podemos remover o mesmo e introduzi-lo proporcionalmente [20].

$$P(Y|X1,...,Xn)\ \alpha\ P(Y)\prod\nolimits_{i=1}^{n} P(Xi|Y)$$

No caso especifico deste problema a classe Y só tem dois valores possíveis, o positivo e o negativo. Em outros casos de estudo é possível que existam mais classes e então as derivações da regra de Bayes serão diferentes [20] pois levarão em conta todas as classes e as máximas probabilidades de pertencer a cada uma delas.

Este algoritmo em específico, disponibilizado pelo azure machine learning studio, aproxima-se daquilo que é o ótimo teórico Bayesiano para classificadores lineares pois escolhe aquele que é o classificador médio [15]. Isto acontece pois num classificador Bayseano o treino iterativo utiliza a distribuição cumulativa dos pesos atribuídos às diferentes métricas para a formação Gaussiana em cada iteração, sendo que esta é levada em conta para a atribuição das restantes métricas. Num classificador Bayes Point Machine, de forma a treinar os pesos de um novo dado de input, é utilizada a média de todos os pesos adquiridos até então, permitindo assim adquirir um classificador médio [37]. Este tipo de classificação permite que o algoritmo seja muito mais resistente a dados que se destaquem dos restantes, sendo que entidades que possuam valores anormais têm menor probabilidade de criar disrupção no sistema.

Uma das vantagens de um classificador que tem como output resultados probabilísticos passa pela compensação a nível da diferença de ocorrências entre classes [33]. Em casos nos quais o número de ocorrências de uma classe seja bastante superior ao de outra, este tipo de classificadores tem em atenção a diferença de ocorrências e escala diferentes inputs de forma a que a classificação obtenha melhores resultados.

6.1.4 Two-class decision forest

Uma árvore de decisão é uma árvore como uma coleção de nós destinada a criar uma decisão sobre a afiliação de valores a uma classe. Cada nó da árvore representa uma regra que divide a árvore para um atributo em especifico, sendo que, para os casos de classificação, esta regra separa os valores pertencentes às diferentes classes [18].

A implementação particular do módulo de azure machine learning utilizado funciona construindo múltiplas árvores de decisão e seguidamente votar naquele que é o output mais popular obtido a partir destas. A utilização do voto é uma das formas mais utilizada para gerar resultados num modelo de classificação [16].

A forma como o algoritmo classifica as entidades pode ser dividida em quatro pontos [16]:

- Diferentes árvores de decisão são criadas, utilizando todo o conjunto de dados, em que o ponto de partida de cada árvore é diferente de árvore para árvore. Este ponto é decidido aleatoriamente, permitindo diferenciação de árvores.

- Cada árvore no conjunto de árvores de decisão produz um output que consiste num histograma não normalizado de frequência para as diferentes classes.
- Um processo de agregação soma estes histogramas e normaliza os resultados, de forma a produzir as probabilidades para cada classe.
- As árvores com uma maior confiança de previsão vêm atribuídos um maior peso na sua classificação para a atribuição final do valor pelo classificador. Sendo que a confiança vai sendo adquirida à medida que a classificação de cada entidade é feita e essa classificação foi acertada.

De forma geral a utilização de árvores trás alguns benefícios, como é o caso de estas possibilitarem captar margens não lineares no que toca às margens que dividem as classes das entidades a classificar, bem como permitem um treino e previsão de grandes quantidades de informação de forma rápida, uma vez que estas baseiam-se num modelo de computação bastante simples [16].

6.1.5 Two-class boosted decision tree

O principio por detrás do *boosting* tem a ver com o facto de se combinarem diferentes classificadores mais fracos de forma a que seja criado um classificador mais estável e com menor erro associado.

No que toca à aplicação direta desde módulo no azure machine learning studio, este baseia-se no principio das árvores de decisão (6.1.4), contudo tem a diferença de que uma segunda árvore corrige os erros da primeira, sendo que uma terceira árvore corrige os erros da segunda e da primeira e assim sucessivamente, até ser alcançado um limite definido. Esta agregação de árvores, apesar de exigir mais do ponto de vista de memória computacional necessária para dar resposta ao problema, tem a vantagem de melhorar a *accuracy* (explicada na Secção 6.2.2) do modelo.

6.1.6 Two-class decision jungle

As selvas de decisão são uma recente extensão das florestas de decisão (Secção 6.1.4), onde uma selva de decisão consiste numa adição de ramos de forma a criar grafos de decisão acíclicos dirigidos, do inglês DAGs (*directed acyclic graphs*). Ao permitir que diferentes ramificações da árvore convirjam para um nó possibilita com que os DAGs tenham uma necessidade de memória menor e uma computação mais rápida do que uma árvore de decisão dita convencional [17].

6.2 Métricas de Avaliação

De forma a perceber que tipo algoritmo possibilita a obtenção de melhores resultados no que toca ao treino dos modelos de aprendizagem automática, é necessário treinar todos

eles em função dos diferentes conjuntos de dados e perceber qual obteve melhor desempenho. Para avaliar corretamente qual o melhor par algoritmo/conjunto de dados fez-se uso de uma série de métricas de avaliação, sendo estas uma matriz de confusão (Secção 6.2.1), a exatidão (Accuracy) descrita na Secção 6.2.2, a precisão (Precision) explicada na Secção 6.2.3, a sensibilidade (Recall) na Secção 6.2.4, a classificação F1 na Secção 6.2.5 e por fim a área por debaixo da curva (Area Under Curve) apresentada na Secção 6.2.6.

6.2.1 Matriz de confusão

Uma matriz de confusão é uma medida de desempenho utilizada para classificadores de aprendizagem automática onde o output dos mesmos pode ter duas ou mais classes, é uma Tabela que possui 4 combinações diferentes de valores onde se relacionam os valores obtidos pela previsão do modelo com os valores reais dos dados.

		Valores verdadeiros	
		Positivos (1)	**Negativos (0)**
Valores previstos	Positivos (1)	VP	FP
	Negativos (0)	FN	VN

Tabela 6.1: Estrutura de uma matriz de confusão

A estrutura de uma matriz de confusão genérica tem o formato apresentado na Tabela 6.1. A matriz de confusão é extremamente útil pois é a partir da informação contida na mesma que se torna possível depois calcular as outras métricas de desempenho, como é o caso do *recall*, *precision*, *accuracy* e a *area under the curve.*

No primeiro quadrante da Tabela encontram-se os verdadeiros positivos (VP), esta informação refere-se à quantidade de entradas classificada positivamente de forma correta, ou seja, na qual o valor original dos dados era 1 e o modelo de classificação atribuiu o valor de 1.

No caso dos verdadeiros negativos (VN), posicionados no último quadrante da Tabela, estes indicam a quantidade de entidades previstas como negativas e que estão corretas, ou seja, todas as entidades classificados com 0 e que são de facto um 0.

Relativamente aos erros existem dois tipos, o falso positivo (FP) é um erro de tipo I, onde o modelo de classificação atribuiu a classe 1 mas na verdade a entidade é da classe 0, estamos por isso na presença de uma classificação sobre algo ter acontecido uma situação de *churn* ou de sugerir algo quando na verdade não aconteceu ou não era necessário. Este tipo de erro leva a que sejam levantados falsos alertas de intervenção para com os clientes, uma vez que são declarados como perigo de *churn* quando na verdade não existe risco associado. Um falso negativo (FN) é um erro de tipo II, este tipo de erro ocorre quando o classificador determina que não existiu uma previsão de *churn* ou necessidade de sugestão quando isso está errado, o output do modelo atribuiu o valor de 0 quando na verdade deveria ser 1. Este tipo de erro é mais grave, pois no contexto do problema abordado nesta dissertação leva a que se perca possibilidades de negócio, perdendo clientes ou

não fazendo o acompanhamento desejado dos mesmos. Estes quatro valores (VP, VN, FP, FN) são a base das métricas de avaliação, sendo que estes serão utilizados de forma a determinar as restantes métricas utilizadas.

6.2.2 Accuracy

A *accuracy* (em português exatidão) mede a fração de previsões que o modelo acertou, ou seja, permite saber a quantidade de valores que obtiveram o valor que deveriam ter tendo em conta o panorama geral. A fórmula de cálculo da accuracy é dada pela divisão de todos os casos classificados corretamente por todos os casos de estudo:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

A *accuracy* sozinha não nos permite extrair conclusões viáveis quando estamos na presença de um conjunto de dados desequilibrado no que toca à distribuição entre as duas classes em estudo, nesses casos é aconselhável utilizar a *precision* e o *recall* para fazer essa avaliação[49].

6.2.3 Precision

A *precision* (em português precisão) representa a habilidade do classificador de não classificar uma entidade negativa como positiva. A fórmula da *precison* é dada pela divisão divisão do total de verdadeiros positivos pela soma dos items que foram classificados como positivos, ou seja, verdadeiros positivos e falsos positivos.

$$Precision = \frac{VP}{VP + FP}$$

6.2.4 Recall

O *recall* (comummente chamado de sensibilidade em português) trata de medir os casos de verdadeiros positivos. O *recall* permite analisar o desempenho do classificador em classificar os exemplos positivo, caso seja do interesse encontrar todos os exemplos positivos então o *recall* deve ser maximizado. Assim sendo a sua fórmula é dada pela divisão dos verdadeiros positivos sobre todos os positivos do conjunto, ou seja, os verdadeiros positivos e os falsos negativos.

$$Recall = \frac{VP}{VP + FN}$$

6.2.5 F1 Score

A classificação F1 combina a *precision* e o *recall* numa única métrica, desta forma torna-se mais fácil comparar dois classificadores. A classificação F1 é a média harmónica entre a *precision* e o *recall*, dada pela seguinte fórmula:

$$F1Score = 2 * \frac{Precision * Recall}{Preicison + Recall}$$

Uma média normal trata todos os valores de igual forma, a média harmónica dá muito mais peso a valores baixos, o resultado é que a classificação F1 apenas apresenta valores elevados se tanto a *precision* como o *recall* tiverem eles mesmos valores elevados, tornando assim a classificação menos suscetível de sofrer mudanças drásticas.

6.2.6 Area Under Curve

A área por debaixo da curva está diretamente ligada a uma métrica de classificação que dá pelo nome de ROC (*Receiver Operator Curve*). A ROC mostra a sensibilidade do classificador através do traçar de uma curva no gráfico que relaciona o rácio de verdadeiros positivos com o rácio de falsos positivos [49].

Se o classificador demonstrar um excelente desempenho então o rácio de verdadeiros positivos vai aumentar, fazendo assim com que a área por debaixo dessa curva seja de aproximadamente 1, caso o classificador tenha um desempenho menos desejado e basicamente classifique as entidades aleatoriamente então a curva vai crescer de forma linear com o aumento de falsos positivos. A Figura 6.4 ilustra os diferentes tipos de AUC e respetivos classificadores associados.

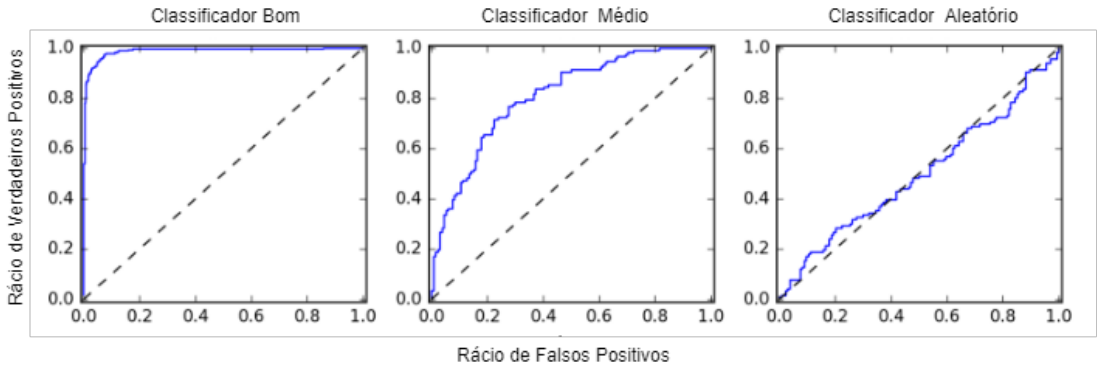


Figura 6.4: Diferentes tipos de AUC e respetivos classificadores

6.3 Etiquetação dos Dados para Treino

Uma vez que está a ser utilizado o método de aprendizagem supervisionada para treinar modelos de aprendizagem automática, é necessário que os dados que são utilizados para o treino dos mesmos apresentem a classe a que pertencem. Esta etiqueta permite com que os algoritmos retirem ilações e percebam de que modo os diferentes valores das métricas influenciam o valor final da etiqueta. Estando a utilizar dados reais relativos aos consumo de serviços de clientes reais, os dados originais não possuem por si só uma etiqueta que indique se aquele cliente está ou não numa situação de *churn*, deste modo é necessário realizar a etiquetação dos dados de acordo com um parâmetro.

No que toca aos dados que serão utilizados para a previsão de casos de *churn*, foi decidido que o parâmetro que define o abandono de um cliente parte do princípio de que se o mesmo não constar nos dados de faturação por um período superior a um mês, então podemos considerar esse cliente como perdido. Esta decisão foi tomada pois no sentido da lógica de negócio este parâmetro enquadrava-se no problema. De forma a etiquetar os dados foi necessário utilizar o módulo de execução de *scripts* em python, disponibilizado pelo azure machine learning studio, com a utilização desta linguagem foi possível executar de forma rápida e eficaz a alteração ao conjunto de dados de forma a incluir a coluna 'Churn'. Esta coluna vai representar a etiqueta que atribui a cada entidade o valor de 0 caso um utilizador não esteja numa situação de *churn* e o valor de 1 caso contrário.

No caso da sugestão de *upgrades* ou *downgrades* de serviços o caso é mais delicado. De forma a ser percetível quando os clientes alteram o seu pacote de serviços foi necessário consultar a matriz de ofertas da Microsoft, onde constam os identificados (ID's) das diferentes subscrições e para que tipo de subscrições estas podem sofrer um *upgrade*. De forma análoga foi necessário criar uma matriz na qual se inverteu a informação contida na matriz de ofertas original de forma a conseguir mapear os possíveis *downgrades* de serviços, assim, com ambas as informações de mapeamento de subscrições disponíveis passou a ser possível percorrer os dados e com base na sua avaliação ao longo do tempo perceber quando os clientes efetuaram mudanças nos seus pacotes. Como um cliente pode efetuar diversos *upgrades* e *downgrades* ao longo tempo, bem como ir alternando entre diferentes pacotes foi necessário criar duas colunas, uma relativa aos *upgrades* e outra relativas à existência de *downgrades*, ambas as colunas funcionam de forma binária, sendo que para cada entrada do cliente no tempo, sempre que existiu uma alteração é atribuído o valor 1 na coluna correspondente, desta forma é possível saber sempre que um cliente efetuou cada *upgrade* e *downgrade* ao longo do tempo.

Uma vez efetuada a etiquetação dos dados passa a ser possível aos modelos de aprendizagem automática utilizarem estas colunas de etiquetas para aprender sobre os dados e assim possibilitar a classificação de entidades.

6.4 Ambiente de testes para modelos de Aprendizagem Automática

De forma a possibilitar a escolha de qual o algoritmo que melhor se adequa aos dados foi necessário montar um ambiente para testar as diferentes possibilidades. A plataforma do azure machine learning studio permite ramificar o *output* providente dos módulos de forma a que este se torne o *input* de múltiplos outros módulos ao mesmo tempo, esta possibilidade permitiu que se testassem os diferentes algoritmos e se comparassem os seus resultados de forma paralela, acelerando assim o processo de testes.

De forma a ser possível treinar e decidir sobre um modelo de aprendizagem automática é necessária a utilização de diferentes módulos, nomeadamente a separação dos dados (6.4.1), treino do modelo (6.4.2), classificação do modelo (6.4.3) e por fim a avaliação do modelo (6.4.4), explicados abaixo.

6.4.1 Separação dos dados

O módulo de separação dos dados, ou em inglês *Split Data*, tem como função dividir o conjunto de dados fornecido em dois conjuntos diferentes. Particularmente útil para utilização na separação de informação para o treino e teste de modelos. A divisão do conjunto de dados dá-se por número de entradas (linhas da Tabela), sendo que é possível definir quantas entradas queremos no conjunto de dados A e no conjunto de dados B. De forma a garantir que os dados dos dois conjuntos apresentam a mesma percentagem de casos de *churn*, por exemplo, é possível ativar a opção de efetuar uma divisão estratificada pela coluna 'Churn', garantindo assim que os casos de *churn* não ficam todos num dos conjuntos, impossibilitando depois a confirmação de que o modelo está a funcionar corretamente. Ao ter a divisão estratificada ativa este módulo divide de igual forma o número de entidades que apresentem casos de *churn* pelos dois conjuntos de saída.

Assim sendo, este módulo apresenta duas saídas de *output*, uma para o conjunto A e outra para o conjunto B resultantes da divisão, o conjunto A será utilizado pelo módulo de treino e o conjunto B pelo módulo de classificação do modelo obtido.

6.4.2 Treino do Modelo

O módulo de treino do modelo, ou em inglês *Train Model*, tem como função efetiva o treino de um modelo de aprendizagem automática, criando relações entre os dados e concluindo sobre os mesmos. Normalmente o treino é composto por três partes, a primeira sendo a escolha do algoritmo em particular que se irá utilizar e a conFiguração dos seus parâmetros, seguidamente o fornecimento de um conjunto de dados já etiquetados e por fim o *output* do módulo indica as diferentes classificações atribuídas às várias métricas do conjunto de dados, de acordo com o peso que estas tiveram para a classificação das entidades [14]. Estas informações são úteis para a classificação de novas entidades deste ponto em diante.

A partir do momento em que o módulo termina a sua execução é possível guardar o *output* do mesmo como um modelo de treino completo, possibilitando assim a sua utilização em previsões futuras. O módulo de *Train Model* permite também que se volte a treinar um modelo de aprendizagem com novos dados, de forma a atualizar o modelo e melhorar os resultados do mesmo. Esta necessidade surge pois ao longo do tempo os padrões entre os clientes podem mudar, assim sendo, de forma a que os modelos continuem atuais é necessário efetuar um novo treino dos mesmos periodicamente de forma a que estes usem dados mais recentes para tirar ilações dos mesmos.

Para classificar novas entidades é necessário que se utilize a junção de dois módulos, conectando assim o output do módulo de *Train Model* ao input do módulo de *Score Data* (Secção 6.4.3), juntamente com o conjunto de dados que desejamos classificar. É justamente o peso a utilizar nas diferentes métricas, informação do algoritmos escolhido, aliado com o conjunto de dados fornecido que torna possível a atribuição de uma classe a cada entrada.

6.4.3 Classificação do modelo

O módulo da classificação do modelo, ou em inglês *Score Data*, possibilita gerar previsões utilizando um modelo já treinado para classificação. Assim que fornecidos os inputs necessários, sendo este o modelo de aprendizagem automática treinado e o conjunto de dados a classificar, torna-se possível para o módulo de *score data* atribuir uma classificação a cada entrada do conjunto de dados, baseada na probabilidade daquela entrada pertencer à classe atribuída [13]. O output do módulo junta ao conjunto de dados fornecido duas colunas, onde consta a classe atribuída e a probabilidade de pertencer à classe atribuída, estas duas colunas serão úteis em dois contextos diferentes. Se estivermos no contexto de analisar a prestação do modelo estas colunas serão analisadas pelo módulo de avaliação do modelo (Secção 6.4.4) de forma a perceber se os resultados de previsão são acertados, se estivemos em contexto de classificação de novas entidades, são estas colunas que efetivamente dão resposta ao problema da classificação.

6.4.4 Avaliação do modelo

O módulo de avaliação do modelo, ou em inglês *Evaluate Model*, avalia os resultados da classificação de entidades providenciado a sua análise de acordo com as métricas descritas na Secção 6.2. Este módulo dá a possibilidade de analisar um único conjunto de resultados ou de analisar dois conjuntos simultaneamente, permitindo assim uma comparação mais rápida e direta entre resultados de diferentes modelos. É por isso um módulo de uso fulcral no que toca à análise de diferentes algoritmos e como estes influenciam o resultados das previsões dos modelos.

Neste ponto o fluxo de módulos deve apresentar-se como consta na Figura 6.5, onde a separação do conjunto de dados, o treino, classificação e avaliação do modelo se encontram interligados, permitindo assim que se meça aquilo que é a prestação do classificador atual em prol dos dados fornecidos.

6.4.5 ConFiguração do Ambiente

De forma a avaliar todos os algoritmos para um dado conjunto de dados, foi montado um ambiente de testes que consiste na utilização dos módulo acima descritos de forma a que seja possível comparar os diferentes algoritmos entre eles.

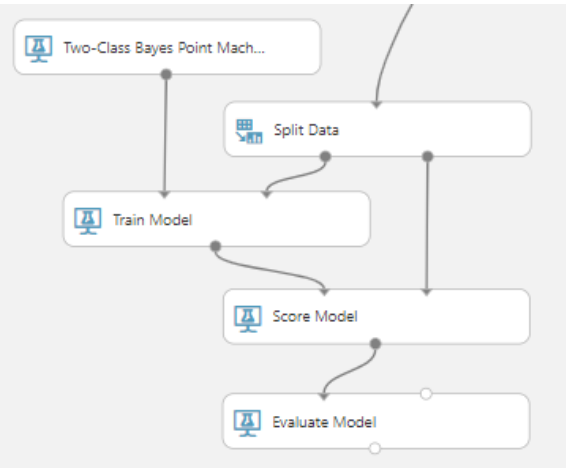


Figura 6.5: Fluxo exemplo de módulos que possibilitam o treino e avaliação de modelos de aprendizagem automática

De modo ser mais elucidativo qual a estrutura do ambiente de testes utilizado, a Figura 6.6 esquematiza tanto a interligação entre os módulos como o fluxo utilizado entre os mesmos.

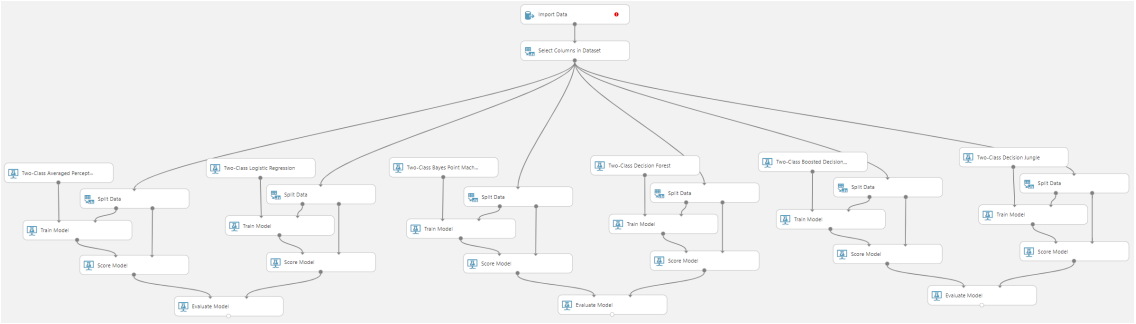


Figura 6.6: Esqueleto de módulos do ambiente de testes para os diferentes algoritmos

Foram utilizadas seis instâncias do conjunto de dados, uma para cada algoritmo em análise, desta forma passa a ser possível testar os algoritmos de forma paralela e no menor espaço de tempo possível. No que diz respeito ao treino particular para cada algoritmo este consiste em separar a informação em dois conjuntos, um para efeitos de treino do modelo e outro para efeitos da classificação, de seguida esta informação é passada para o módulo de treino do modelo, onde o modelo de aprendizagem automática é treinado, o *output* deste módulo é então utilizado como *input* do módulo de classificação. Como o módulo de avaliação de modelos permite que se comparem os resultados de dois classificadores ao mesmo tempo, este foi utilizado de forma a comparar os resultados dos algoritmos dois a dois.

Com este ambiente montado passa a ser possível correr todos os algoritmos numa única execução, treinar assim os diferentes modelos e perceber de que forma os diferentes algoritmos influenciam o resultado final para cada conjunto de dados.

6.5 Sumário

Neste Capítulo são abordados os diferentes algoritmos utilizados, métricas de avaliação de desempenho e configuração do ambiente de testes, sendo que todos estes pontos são vitais para que se faça uma análise correta de que combinação de algoritmos e conjunto de dados melhor se adequam ao problema desta dissertação. A comparação de resultados provenientes dos diferentes modelos treinados na plataforma Azure Machine Learning Studio encontra-se presente no Capítulo 7.

Capítulo 7

Teste e Treino de Modelos de Aprendizagem Automática

Neste capítulo é abordado o treino e teste dos vários modelos de aprendizagem automática.

Mais uma vez, à semelhança do que foi feito na Secção 5.2), o teste dos diferentes algoritmos, análise das métricas classificativas de resultados e o treino efetivo dos modelos de aprendizagem automática encontra-se dividido em duas partes, a primeira relativa aos dados de subscrições (Secção 7.1) e a segunda relativa aos dados de utilização (Secção 7.2).

Este Capitulo termina com as conclusões retiradas dos testes efetuados e com a escolha de qual o melhor modelo de aprendizagem automática a utilizar (Secção 7.3).

7.1 Testes para Dados de Subscrições

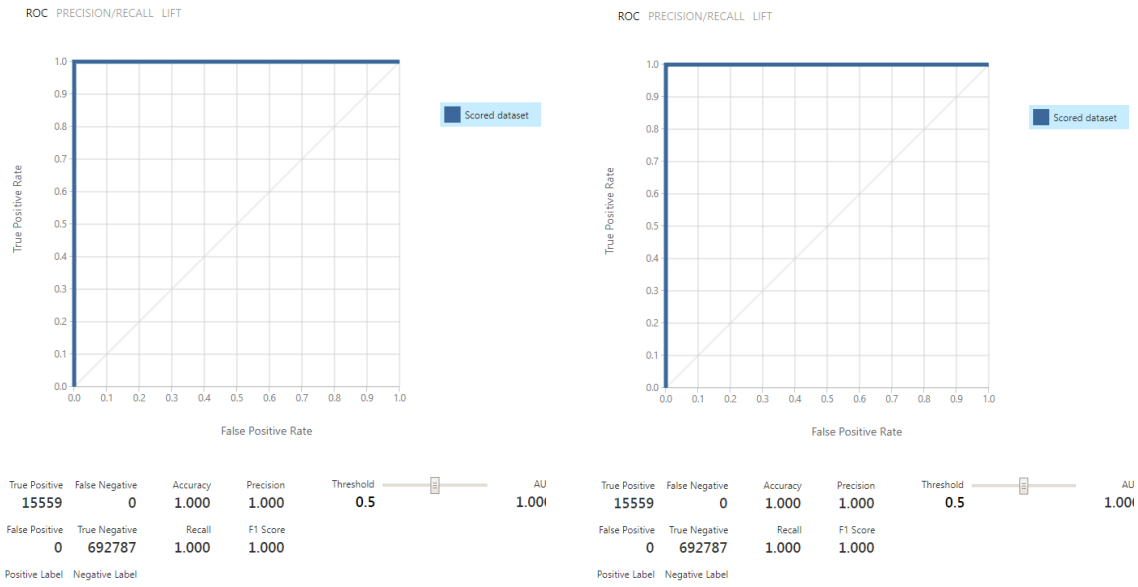
Relativamente aos dados de subscrições foram efetuados testes sobre o desempenho de todos os modelos treinados com os algoritmos apresentados na Secção 6.1 para todos as agregações de dados criadas na Secção 5.2.1.1. Este tipo de dados permitiu que fossem implementados modelos de aprendizagem automática para deteção de situações de *churn* (Secção 7.1.1) bem como para previsão de situações em que se deve sugerir pacotes de serviços aos clientes (Secção 7.1.2).

7.1.1 Teste de algoritmos para classificação de churn

De forma a avaliar todas as possibilidades e mostrar os resultados de forma consistente, serão realizados testes a todas as agregações de dados e em todas elas serão apresentados os resultados pela ordem de algoritmos apresentada na Secção 6.1.

7.1.1.1 Agregação por cliente em função do tempo

No que toca à agregação por cliente, esta obteve resultados excelentes tanto para o algoritmo de *averaged perceptron* como para o algoritmo de *logistic regression*. Como é possível visualizar na Figura 7.1 ambos os testes apresentam um valor de área por debaixo de curva (AUC) de 1, e é possível ver que a matriz de confusão apresenta um total de zero falsos negativos e zero falsos positivos, o que significa que todas as entidades que se classificaram foram efetivamente classificadas corretamente.



(a) Averaged perceptron (b) Logistic regression

Figura 7.1: Métricas de desempenho dos algoritmos de Averaged Perceptron e Logistic Regression para a agregação por cliente em função do tempo

Uma vez que a *accuracy* do modelo é alta, existe a possibilidade do modelo estar a realizar *overfitting* aos dados, assim sendo foi necessário realizar testes adicionais de forma a perceber se os resultados obtidos eram de facto fruto de *overfitting* ou se o modelo estava a funcionar corretamente com resultados excelentes. Para tal recorreu-se à utilização de um módulo disponibilizado pelo Azure Machine Learning Studio, o *Cross Validate Model*.

A forma como o *Cross Validate Model* funciona garante o teste de modelos com informação nunca antes processada pelos mesmos, isto verifica que os modelos não estão a memorizar informações dos dados nem se estão a ajustar aos mesmos, pois o modelo é testado com diferentes sub-conjuntos do conjunto inicial. Este módulo cria 10 sub-conjuntos, sensivelmente do mesmo tamanho, e testa o modelo treinado para cada um dos sub-conjuntos individualmente, testando as métricas de avaliação *Accuracy*, *Precision*, *Recall*, *F1-Score*, *AUC*, mostrando assim o valor para cada sub-conjunto.

Na Tabela 7.1 são apresentados os resultados obtidos por parte do módulo de *Cross Validate Model*, onde é visível que os elevados valores de *accuracy* e AUC são transversais aos diferentes sub-conjuntos. Numa situação de *overfitting* seria de esperar que ao testar

com sub-conjuntos mais pequenos e totalmente diferentes os valores variassem muito e a *accuracy* passasse a ter um valor muito mais baixo, como os resultados indicam o contrário podemos então concluir que os resultados obtidos na Figura 7.1 dizem respeito a dados corretos sem a presença de *overfitting*.

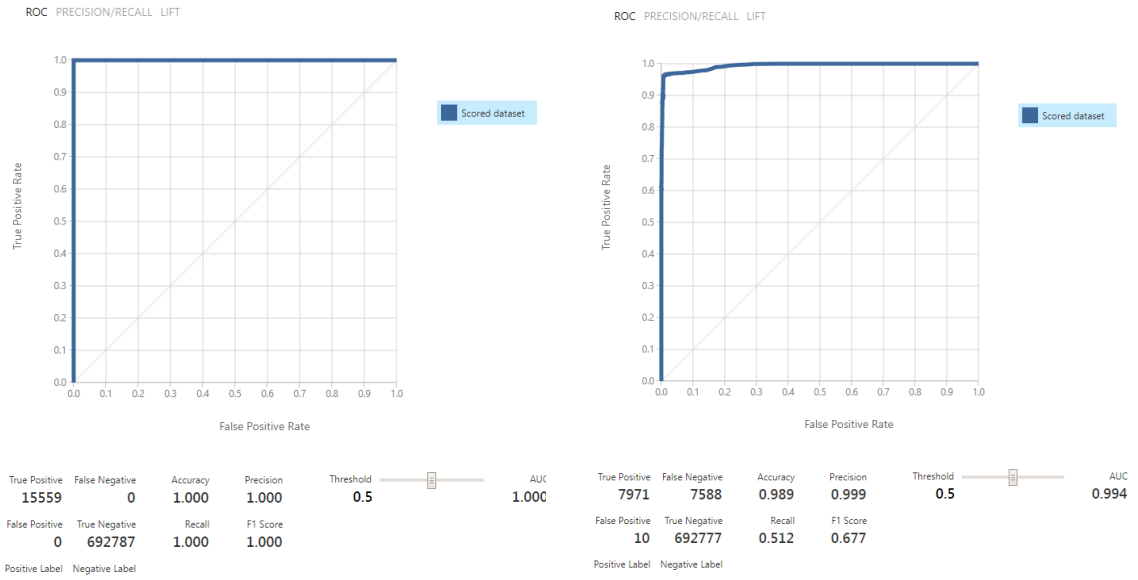
Sub-conjunto	Accuracy	Precision	Recall	F1-Score	AUC
0	0.983906	0.670132	0.535292	0.59517	0.921929
1	0.985854	0.748305	0.556045	0.638006	0.923768
2	0.984993	0.705433	0.537169	0.609908	0.919042
3	0.984492	0.688463	0.532882	0.600763	0.924512
4	0.98529	0.719759	0.538486	0.616065	0.927994
5	0.984358	0.702303	0.53375	0.606534	0.921294
6	0.984062	0.69274	0.536502	0.604692	0.922711
7	0.984845	0.673511	0.548128	0.604385	0.924564
8	0.98529	0.718882	0.54458	0.619708	0.926725
9	0.985311	0.686413	0.555854	0.614272	0.922513

Tabela 7.1: Matriz output do módulo de *Cross Validate Model*

A Figura (7.2), apresentada abaixo, mostra apresentam-se em (a) os resultados referentes à avaliação do algoritmo Two-Class Bayes Point Machine e em (b) os resultados para o algoritmo Two-Class Decision Forest.

Relativamente à Figura 7.2(a) os resultados foram tão bons como para os algoritmos apresentados anteriormente na Figura 7.1, contudo no caso da Figura 7.2(b) os resultados já não são tão bons. A existência de uma grande quantidade de falsos negativos demonstra que o algoritmo classificou diversas entidades como não tendo risco de *churn* quando isto não era verdade, num contexto de aplicação ao problema real, isto significa que se iriam perder clientes sem que antes estes fossem sinalizados e fossem iniciados os mecanismos necessários à tentativa de reter essesclientes. Podemos por isso dizer que o resultado proveniente do uso do algoritmo de Two-Class Decision Forest foi inferior aos restantes até à data para esta agregação de dados.

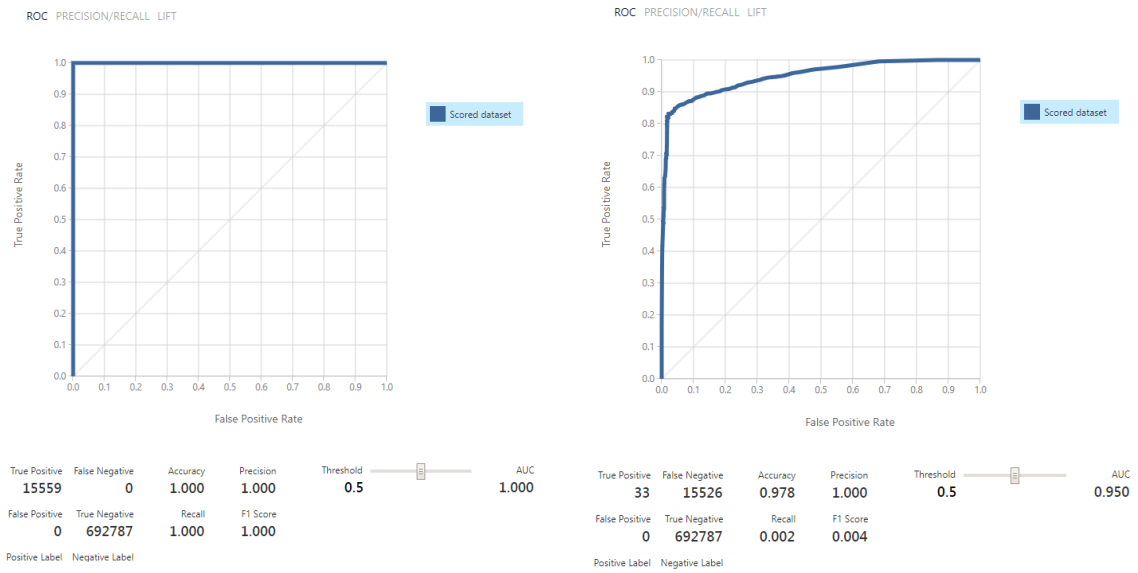
Relativamente aos dois últimos algoritmos utilizados, sendo eles o algoritmo de Boosted Decision Tree e Decision Jungle, estes foram avaliados da mesma forma que os restantes e os dados de avaliação encontram-se ilustrados na Figura 7.3. Relativamente aos resultados de Boosted Decision Tree estes tiveram resultados perfeitos, tal como já aconteceu com a utilização de outros algoritmos, aparentemente a utilização da agregação de dados em questão juntamente com o problema de responder à questão dos *churns* permite que esta seja respondida de igual forma por diferentes algoritmos. Relativamente à utilização do algoritmo de decision jungle, apresentado na Figura 7.3(b) este teve um baixo desempenho, caracterizado pela existência de baixa quantidade de entidades classificadas como verdadeiros positivos e um número muito elevado de falsos negativos. O que significa que o algoritmo raramente atribuiu o valor de perigo de *churn* a uma entidade, deixando assim passar todos os casos em que se deveria atuar.



(a) Bayes Point Machine

(b) Decision Forest

Figura 7.2: Métricas de desempenho dos algoritmos de Bayes Point Machine e Decision Forest para a agregação por cliente em função do tempo



(a) Boosted Decision Tree

(b) Decision Jungle

Figura 7.3: Métricas de desempenho dos algoritmos de Boosted Decision Tree e Decision Jungle para a agregação por cliente em função do tempo

7.1.1.2 Agregação por cliente e subscrição em função do tempo

No que toca à agregação por cliente, esta obteve resultados muito bons tanto para o algoritmo de *averaged perceptron* como para o algoritmo de *logistic regression*. De frisar que apenas a linha vermelha no gráfico 7.4(a) e a linha azul do gráfico 7.4(b) devem ser tidas em consideração na Figura 7.4, pois são essas linhas que dizem respeito ao

algoritmo em estudo, a outras linhas presentes no gráfico dizem respeito a outros testes que estavam a ser comparados com o gráfico em questão, contudo a visualização dentro do Azure Machine Learning Studio não permite ocultar uma das linhas.

Quando olhamos para as métricas como *accuracy*, *recall*, *precision e F1 Score* de ambos os resultados dos algoritmos as diferenças não são muitas, contudo, no que diz respeito à matriz de confusão a existência de falsos positivos na Figura 7.4(a) é algo que difere para a Figura 7.4(b), um falso positivo acontece quando é atribuída a noção de possível *churn* a um cliente que na verdade não está em perigo, do ponto de vista do negócio é preferível este tipo de erro do que deixar o cliente passar sem alerta e perder-se esse mesmo cliente, contudo, uma vez que ambos os modelos têm uma percentagem semelhante de falsos negativos a utilização do algoritmo de averaged perceptron acaba por ser mais vantajosa, pois permite economizar tempo e recursos que de outra forma seriam gastos a combater uma situação que não necessitava de ser combatida, por ser um falso alarme.

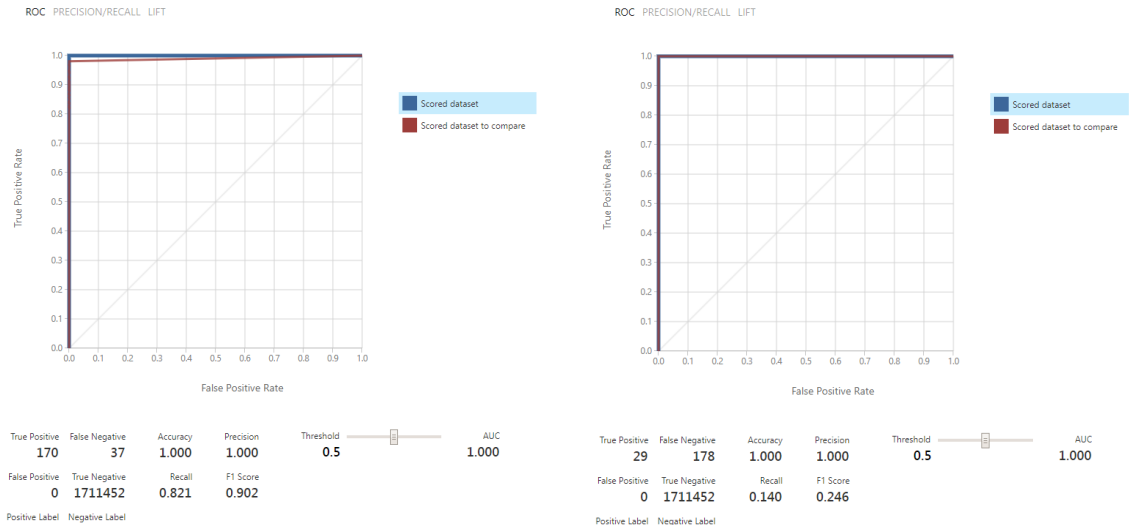
(a) Averaged Perceptron

(b) Logistic Regression

Figura 7.4: Métricas de desempenho dos algoritmos de Averaged Perceptron e Logistic Regression Jungle para a agregação por cliente e subscrição em função do tempo

Passando agora para a análise dos algoritmos de Bayes Point Machine, Figura 7.5(a) e Decision Florest, ilustrado na 7.5, foi possível obter valores bastantes bons no que toca a AUC para ambos os algoritmos, contudo apesar de ambos apresentarem um valor de *accuracy* e *precision* igual, os valores tanto de *recall* como de *F1 Score* são bastante diferentes. Esta diferença fica a dever-se à diferença de verdadeiros positivos apresentados pelos dois modelos, como é possível verificar, o número de verdadeiros positivos obtidos com o uso do algoritmo de Decision Forest foi bastante inferior, esta fraca classificação leva a que, ao aplicar o algoritmo aos dados reais do problema, se possa incorrer no risco de perderem muitos clientes por falta de aviso, uma vez que as entidades que deveriam ter sido classificadas como verdadeiros foram classificadas como falsos, constando por isso no quadrante dos falsos negativos. Assim sendo, de entre os dois algoritmos representados

na Figura é preferível a utilização do algoritmo de *Bayes Point Machine* para endereçar o domínio do problema abordado nesta dissertação quando comparado com o algoritmo de *Decision Forest*.



(a) Bayes Point Machine

(b) Decision Forest

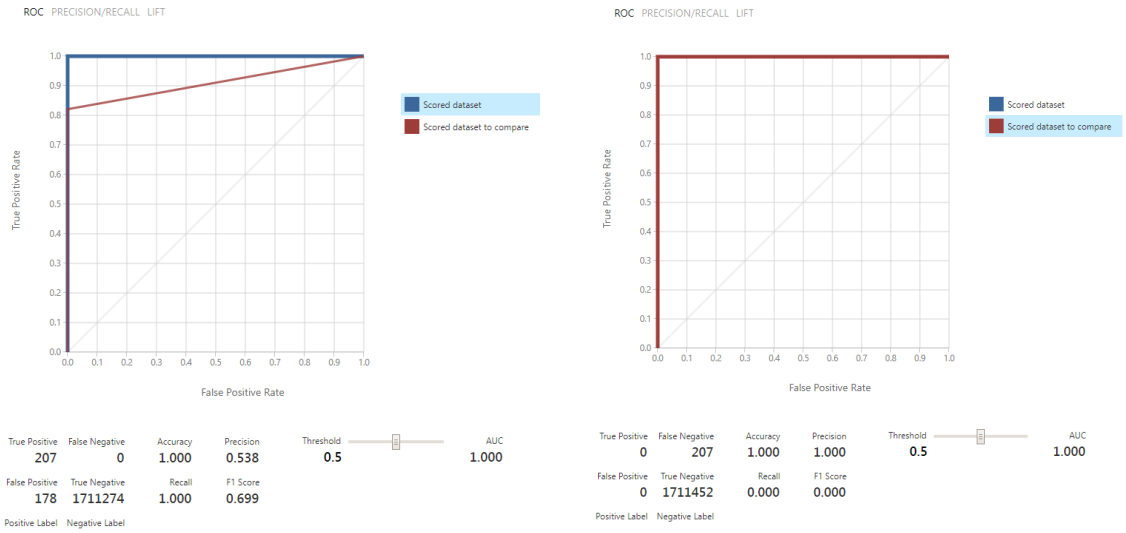
Figura 7.5: Métricas de desempenho dos algoritmos de Bayes Point Machine e Decision Forest para a agregação por cliente e subscrição em função do tempo

No que toca à análise dos últimos dois algoritmos para esta agregação de dados, cujos resultados são apresentados na Figura 7.6, os resultados presentes na Figura 7.6(a) permitem verificar que não existe a existência de nenhum caso de falso negativo (ver linha azul), o que é bom, pois significa que todos os clientes que estavam em perigo de deixar de ser clientes foram efetivamente identificados. Existem alguns falsos positivos, pelo que irão ser sinalizadas situações que não necessitavam de intervenção, contudo, do ponto de vista do negócio é preferível ter um maior controlo e sinalização do que perder um cliente, pelo que o resultado da aplicação do algoritmo faz todo o sentido do ponto de vista do negócio.

Relativamente ao algoritmo de decision jungle, cujos resultados se encontram na Figura 7.6(b), todas as entidades tiveram um valor atribuído de zero, ou seja, como não se encontrando em perigo de *churn*. A ausência de verdadeiros positivos e falsos positivos demonstra que o algoritmo não conseguiu estabelecer a relação entre os dados que permitisse classificar as entidades entre as duas classes, sendo que com o algoritmo de decision jungle não foi possível classificar uma única entidade corretamente para o caso de *churn*, podemos desde já excluir este algoritmo como candidato à utilização do modelo final, dado que este não satisfaz as necessidades do domínio de aplicação.

7.1.1.3 Agregação pela métrica Date, Cliente + Produto

Tal como referido na Secção 5.2.1.1 existem duas agregações que levam em consideração a coluna Date para formar o seu conjunto de dados, sendo que uma delas tem em conta o



(a) Boosted Decision Tree

(b) Decision Jungle

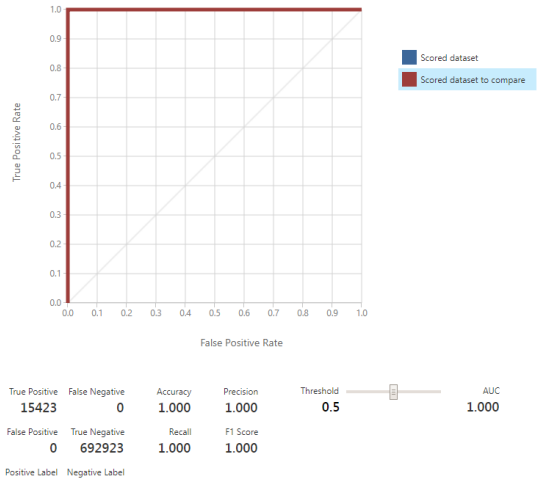
Figura 7.6: Métricas de desempenho dos algoritmos de Boosted Decision Tree e Decision Jungle para a agregação por cliente e subscrição em função do tempo

cliente e o produto da subscrição e outra vai mais fundo nos dados, aumentando o grau de detalhe dos dados adicionando a informação relativa ao serviço. Neste sentido, começando pelos dados que incluem apenas os dados agregador por cliente e produto temos os valores presentes na Figura 7.7, referentes aos algoritmos de Averaged Perceptron e Logistic Regression.

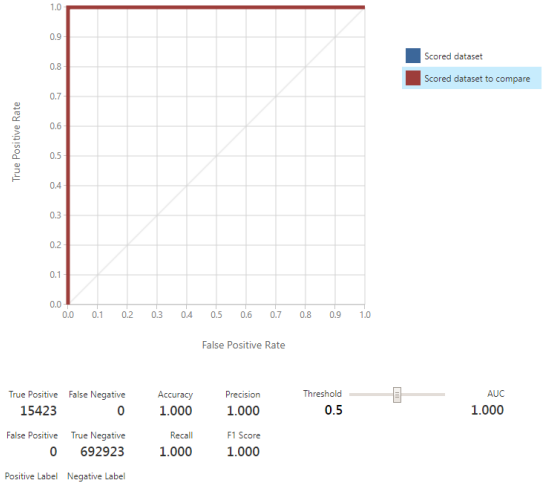
Tal como já havia acontecido para outras agregações de dados tanto o uso de um algoritmo como do outro revelou muito bons resultados, neste caso especifico da agregação por cliente e produto os resultados de ambos são exatamente os mesmos, pelo que podemos atribuir o mesmo grau de eficácia a ambos os modelos. Uma vez que ambos os modelos apresentam valores de classificação excelentes, estes tornam-se candidatos óbvios a serem selecionados para utilização no modelo final do problema.

Passando à análise de mais dois algoritmos para este conjunto de dados, a Figura 7.8(b), ilustra os resultados para o algoritmo de decision forest, que apresenta um elevado número de falsos negativos, este valor leva a que o *recall* e *F1 score* do modelo sejam mais baixos do que o ideal, estes fatores levam à conclusão de que a aprendizagem do modelo por via deste algoritmo não seja a melhor, pelo que podemos deixar de considerar este como uma opção adequada para o uso no projeto. Relativamente ao algoritmo treinado com o algoritmo de bayes point machine, cujos resultados estão na Figura 7.8(a), teve um desempenho tão bom como o demonstrado pelos algoritmos analisados na Figura 7.7, sendo por isso uma opção viável na implementação final.

A análise dos algoritmos boosted decision tree e decision jungle encontra-se expressa na Figura 7.9, ambas em linhas vermelhas. O primeiro aspeto que salta á vista é a curva obtida pelo modelo ilustrado na Figura 7.9(b), relativo ao desempenho do decision jungle. Este algoritmo mostrou-se inadequado para dar resposta ao problema em questão. De

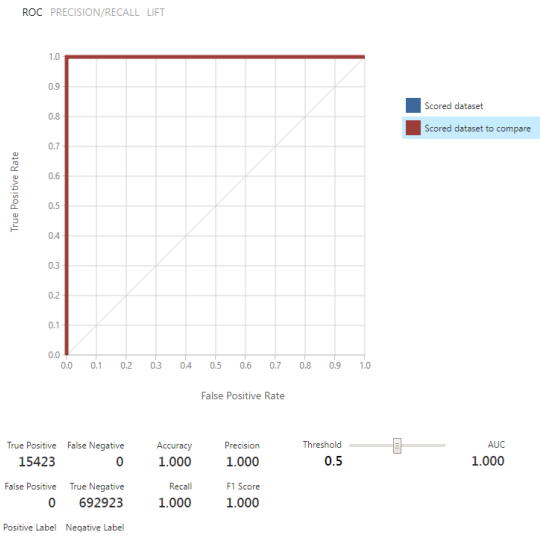


(a) Averaged Perceptron

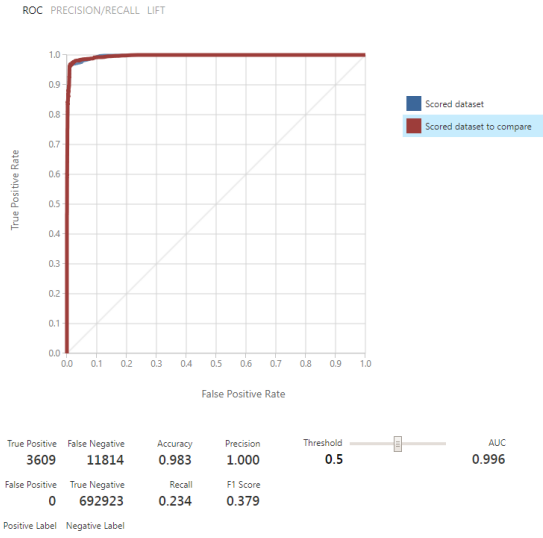


(b) Logistic Regression

Figura 7.7: Métricas de desempenho dos algoritmos de Averaged Perceptron e Logistic Regression para a agregação de cliente e produto, pela métrica Date



(a) Bayes Point Machine

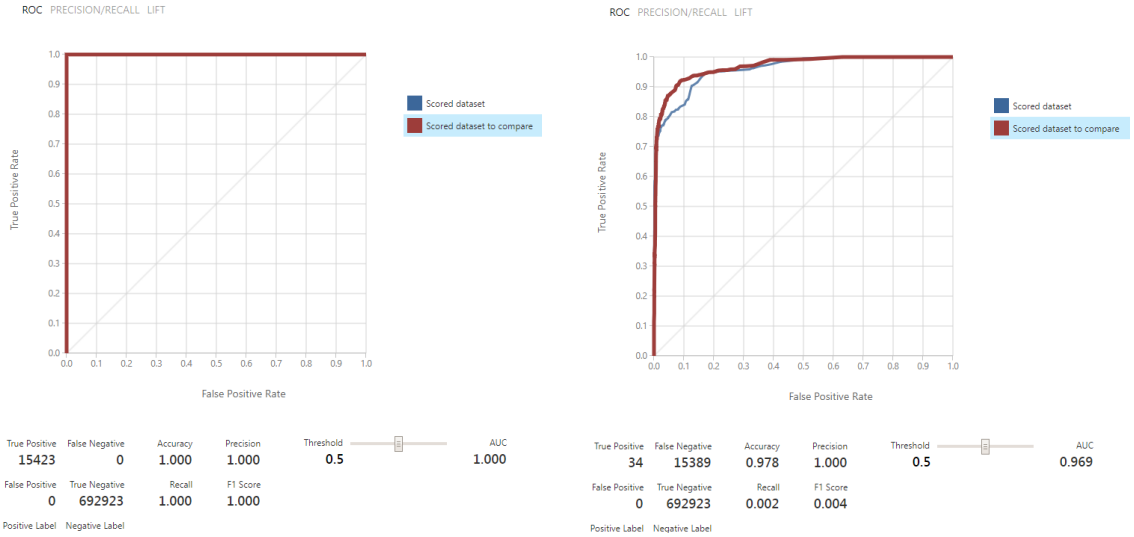


(b) Decision Florest

Figura 7.8: Métricas de desempenho dos algoritmos de Bayes Point Machine e Decision Florest para a agregação de cliente e produto, pela métrica Date

facto foi possível classificar algumas entidades corretamente no que diz respeito a verdadeiros positivos, contudo, quando comparado com o número de falsos positivos obtido o algoritmo não teve o desempenho que deveria, ao aplicar este algoritmo na solução do problema este apenas assinalaria uma pequena fração dos clientes que estão efetivamente em perigo de deixar de ser clientes, estas condições não são de todo as mais favoráveis, principalmente pelo facto de já ter sido possível adquirir combinações de conjuntos de dados e algoritmos que demonstraram resultados substancialmente melhores. No que

toca aos resultados do algoritmo de boosted decision tree, estes mostram que este algoritmo é um bom candidato, uma vez que obteve métricas de avaliação tão boas quanto os melhores algoritmos testados e analisados até ao momento.



(a) Boosted Decision Tree

(b) Decision Jungle

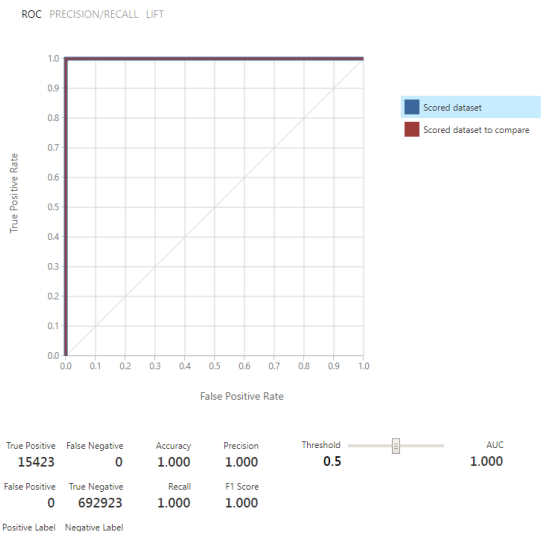
Figura 7.9: Métricas de desempenho dos algoritmos de Boosted Decision Tree e Decision Jungle para a agregação de cliente e produto, pela métrica Date

7.1.1.4 Agregação pela métrica Date, Cliente + Produto + Serviço

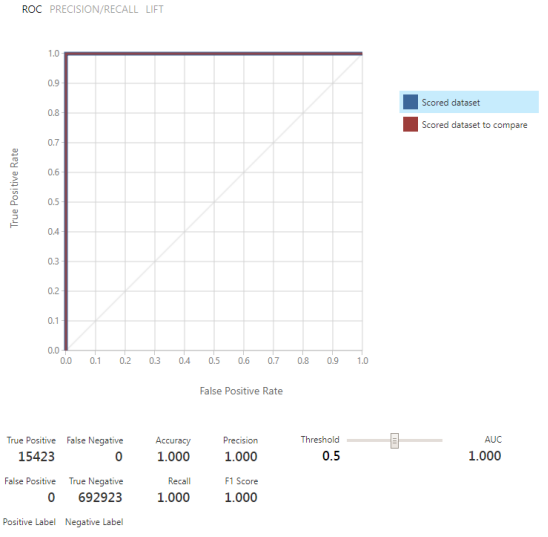
A agregação em estudo é semelhante à utilizada na secção anterior, sendo que tem a diferença de tambem considerar a informação do serviço especifico, começando uma vez mais pelos algoritmos de averaged perceptron e logistic regression, cujos resultados se encontram presentes em 7.10, estes demonstram uma vez mais resultados excelentes do ponto de vista de previsão de resultados, uma vez que não existem casos de falsos positivos nem de falsos negativos.

Na Figura 7.11 constam os resultados para os algoritmos de bayes point machine e de decision forest, tal como já foi verificado em outras agregações de dados cujo objetivo é também a previsão de situações de *churn* o algoritmo de bayes point machine tem uma prestação igual à dos algoritmos de averaged perceptron e logistic regression. No que diz respeito ao desempenho do algoritmo de decision forest, apesar apresentar bons resultados, no contexto de um classificador, por mais uma vez demonstrar um desempenho inferir aos demais, este acaba por ficar excluído de candidato a ser utilizado no modelo final da solução.

A Figura 7.12 permite concluir que, no que toca às agregação com base da coluna Date, a utilização da informação do serviço não foi decisiva no que toca à qualidade dos resultados obtidos, pois para todos os algoritmos utilizados, quer estejamos a considerar os dados com ou sem os dados específicos do serviço os resultados foram praticamente

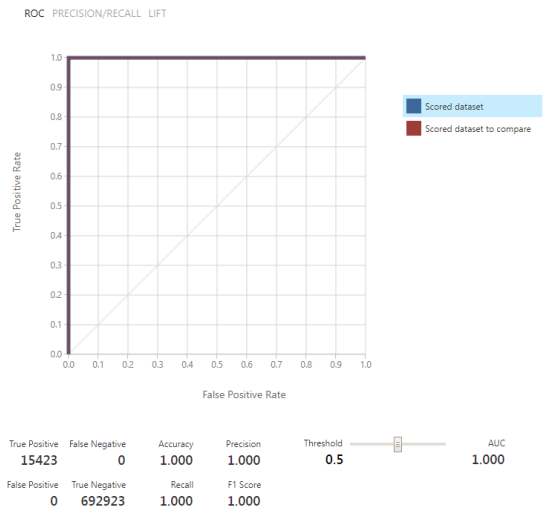


(a) Averaged Perceptron

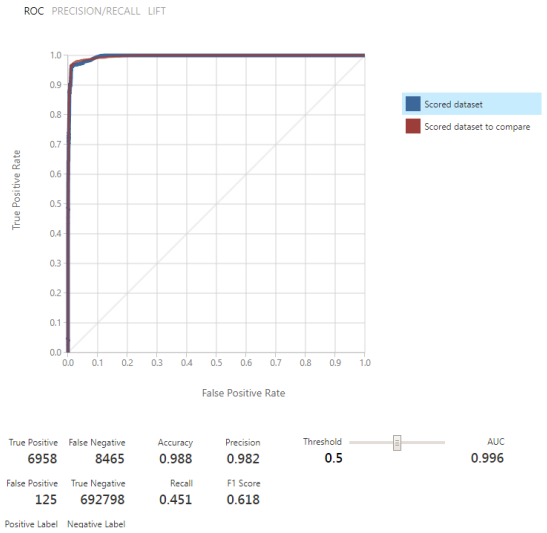


(b) Logistic Regression

Figura 7.10: Métricas de desempenho dos algoritmos de Averaged Perceptron e Logistic Regression para a agregação de cliente, produto e serviço, pela métrica Date



(a) Bayes Point Machine



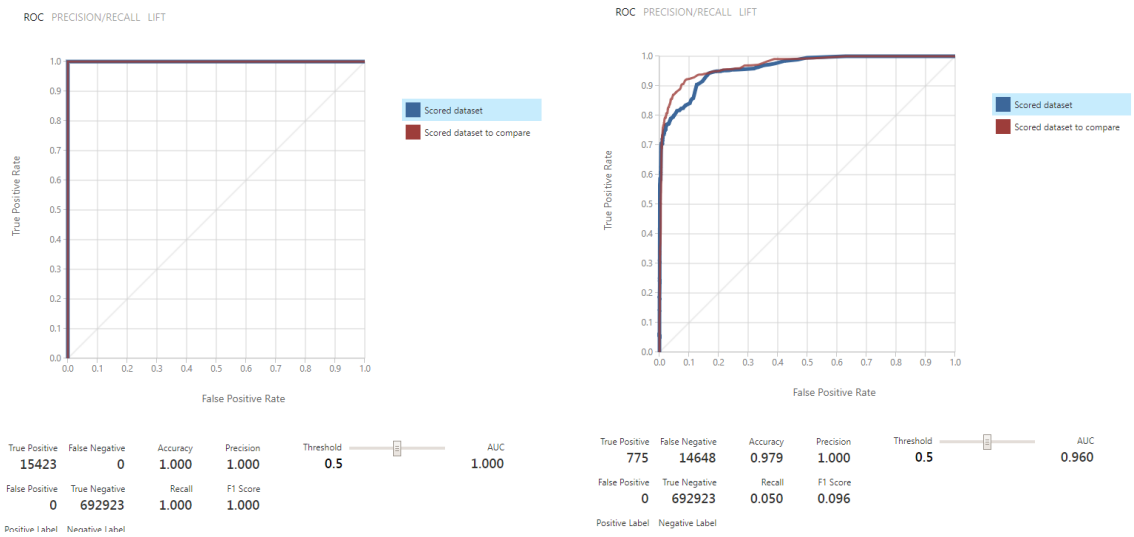
(b) Decision Florest

Figura 7.11: Métricas de desempenho dos algoritmos de Bayes Point Machine e Decision Florest para a agregação de cliente, produto e serviço, pela métrica Date

os mesmos. Mostrando assim que para as duas agregações com base da coluna Date os algoritmos que apresentam o melhor desempenho para a classificação das entidades em risco de *churn* são os algoritmos de averaged perceptron, logistic regression, bayes point machine e boosted decision tree.

Uma vez que ambas as agregações de dados realizadas apresentaram resultados semelhantes, o uso de diferentes granularidades dos dados em nada veio alterar os resultados finais dos classificadores para os algoritmos mencionados acima. Esta informação é útil do ponto de vista de negócio pois levanta questões sobre até que ponto certas informações

devem ou não ser levadas em conta no que diz respeito à faturação dos produtos.



(a) Boosted Decision Tree (b) Decision Jungle

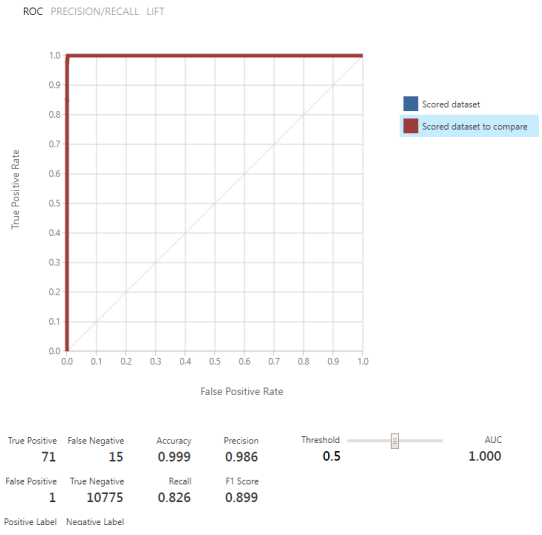
Figura 7.12: Métricas de desempenho dos algoritmos de Boosted Decision Tree e Decision Jungle para a agregação de cliente, produto e serviço, pela métrica Date

7.1.1.5 Agregação pela métrica ProcessedDateTime, Cliente + Produto

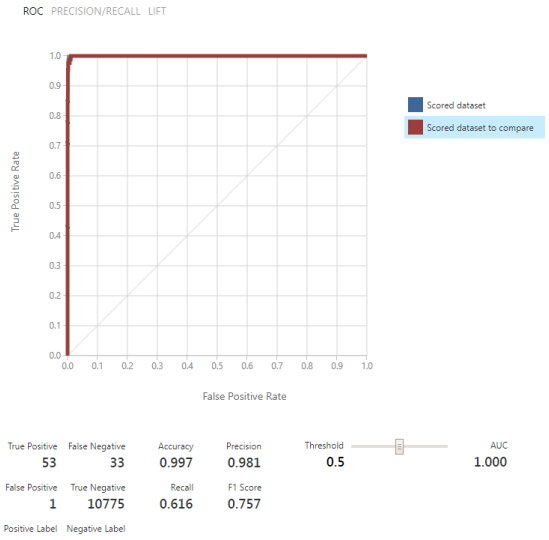
As seguintes duas agregações em teste têm por base a coluna ProcessedDateTime, que tem menos valores presentes do que a coluna Date, pelo que as agregações dos dados são maiores, produzindo assim um total de menos entradas de dados.

Na Figura 7.13 são apresentadas as análises para os algoritmos de averaged perceptron e logistic regression. No que toca a parâmetros como a AUC ambos os gráficos apresentam valores excelentes e como classificadores os valores de *accuracy*, *precision*, *recall* e *f1 score* apresentam bons resultados, contudo, quando comparados com os resultados obtidos pelos mesmos algoritmos com outras agregações de dados estes demonstram-se um pouco abaixo daquilo que tem sido o normal, uma vez que apresentam entidades classificadas erradamente, como é possível verificar pelos falsos negativos e falsos positivos, pelo que, a utilizar estes algoritmos seria mais útil utilizar os mesmos mas aliados a outros conjuntos de dados.

Os dados relativos aos algoritmos de bayes point machine e de decision forest encontram-se expressos na Figura 7.14. Estes dados são interessantes do ponto de vista experimental pois foi a primeira vez que estes algoritmos apresentaram curvas (neste caso expressas a vermelho) deste tipo. Relativamente ao primeiro algoritmo na Figura 7.14(a) este teve um desempenho bastante abaixo das alternativas quando comparado com as demais prestações para os outros conjuntos de dados, a falta de verdadeiros positivos e falsos positivos demonstram que o algoritmos não conseguiu atribuir a classificação de 1 a nenhuma entidade, provando assim que ao não conseguir prever uma situação de *churn*. Relativamente ao gráfico na Figura 7.14(b) este apresenta resultados bastante melhores à primeira vista,



(a) Averaged Perceptron



(b) Logistic Regression

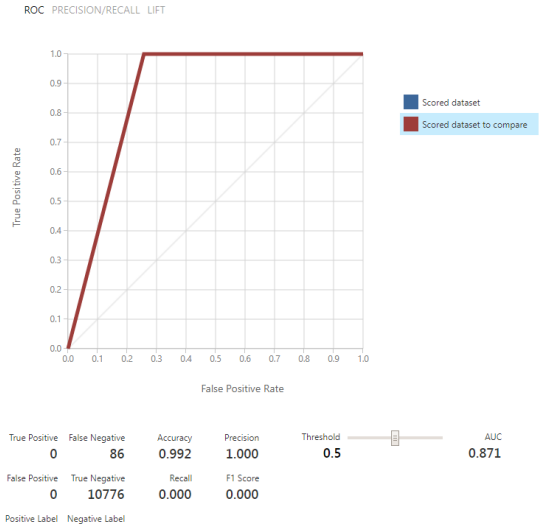
Figura 7.13: Métricas de desempenho dos algoritmos de Averaged Perceptron e Logistic Regression para a agregação de cliente e produto, pela métrica ProcessedDateTime

contudo, uma vez que este também não apresenta nenhuma entidade classificada como potencial perigo de *churn*, também este algoritmo de decision forest apresenta resultados que não permitem dar resposta ao problema da previsão de *churn*.

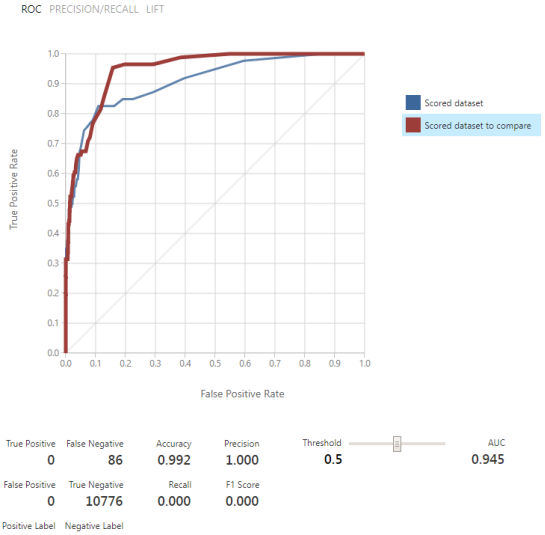
Neste momento este tipo de resultados levam à conclusão de que efetivamente as diferentes combinações entre agregações e algoritmos produzem resultados significativamente diferentes, chegando mesmo a mostrar que o mesmo algoritmo pode tanto dar resultados excelentes como resultados que não correspondem às expectativas, alterando apenas o conjunto de dados fornecido.

No que toca à análise dos últimos dois algoritmos estes revelaram resultados consistentes com outras experiências, como é visível na Figura 7.15, o algoritmo de boosted decision tree tem demonstrados maioritariamente bons resultados quando aplicado, e com o uso desta agregação de dados em especifico isso não se alterou. Apresentando uma AUC perfeita o algoritmo apresenta apenas um total de 3 falsos positivos, a classificação destes três falsos alarmes não é grave, pelo que podemos atribuir uma classificação alta ao resultado do classificador, contudo, já foi possível adquirir resultados mais promissores no que toca à utilização do algoritmos de boosted decision tree, pelo que a utilizar este algoritmos seria uma melhor opção usa-lo em união com outra agregação de dados.

Para o gráfico espesso em na Figura 7.15(b) devemos considerar a linha em vermelho, apesar da linha indicar uma boa previsão a matriz de confusão mostra que apenas foi possível classificar corretamente entidades que não apresentam risco de *churn*, mais uma vez, semelhante ao que tem acontecido com outros classificadores, a utilização de um classificador com estes resultados não manifesta o cumprimento dos objetivos, pelo que podemos automaticamente excluir esta combinação de algoritmos e conjunto de dados do conjunto de hipóteses a considerar.

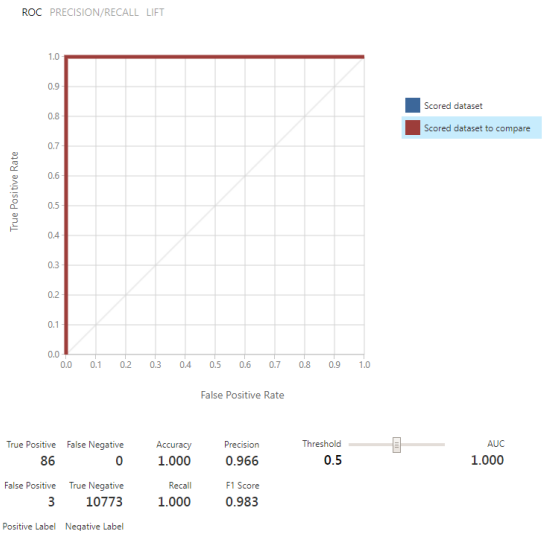


(a) Bayes Point Machine

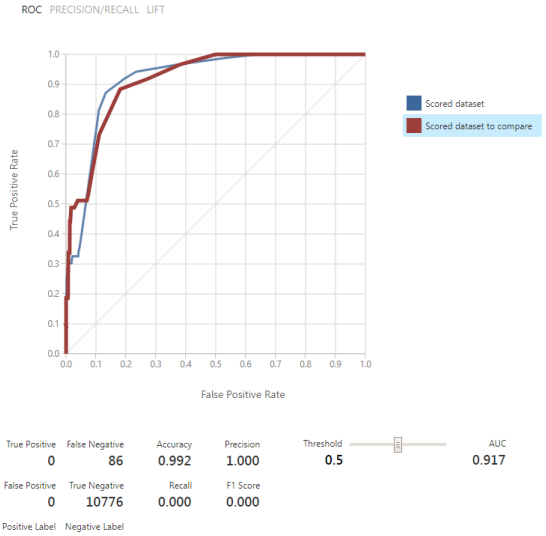


(b) Decision Florest

Figura 7.14: Métricas de desempenho dos algoritmos de Bayes Point Machine e Decision Florest para a agregação de cliente e produto, pela métrica ProcessedDateTime



(a) Boosted Decision Tree



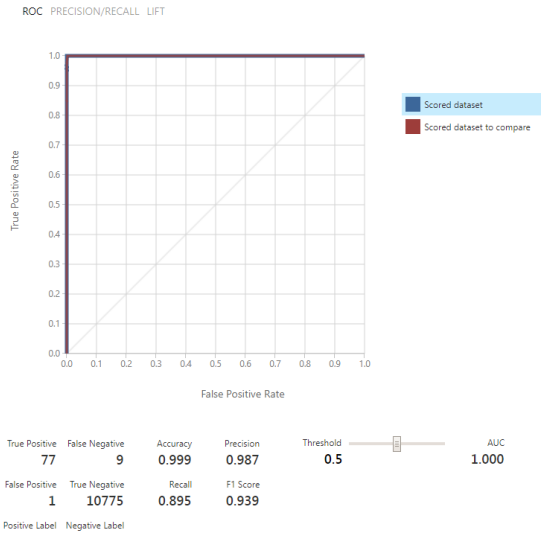
(b) Decision Jungle

Figura 7.15: Métricas de desempenho dos algoritmos de Boosted Decision Tree e Decision Jungle para a agregação de cliente e produto, pela métrica ProcessedDateTime

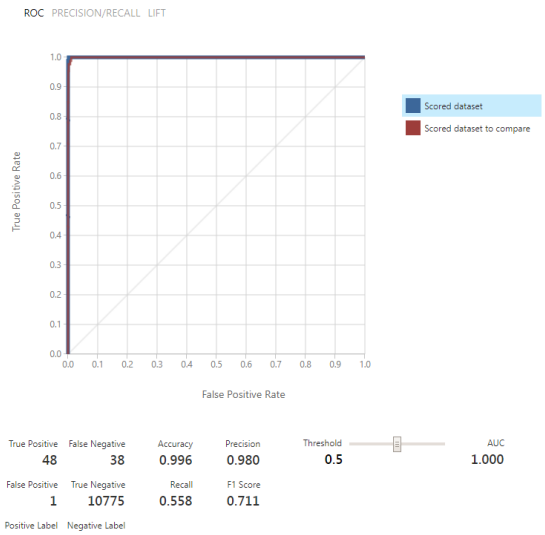
7.1.1.6 Agregação pela métrica ProcessedDateTime, Cliente + Produto + Serviço

A agregação em estudo nesta secção apenas difere da anterior no que toca aos dados de serviços, permitindo assim ter uma abordagem que considera os dados de utilização de serviços agregados por cliente, produto consumido e serviço utilizado A Figura 7.16 mostra os resultados da avaliação para os algoritmos de averaged perceptron e logistic regression, estes dois algoritmos têm conseguido valores de avaliação consistentemente bons ao longo dos diversos testes para os dados de subscrições para previsão das situações

de *churn*. Mais uma vez os resultados na Figura 7.16(a) como na Figura 7.16(b) os resultados provam a presença de dois classificadores bom, apenas com um número reduzido de entidades a serem colocadas no quadrante dos falsos positivos, contudo, no que diz respeito ao algoritmo de logistic regression em particular este apresentou um número consideravelmente maior de falsos negativos do que o de average perceptron. Este tipo de erro, tal com já foi mencionado, é bastante perigoso, pois ao classificarmos uma entidade com o valor de zero quando na verdade esta era um caso de potencial perda de cliente pode conduzir a que se perca efetivamente o cliente e nada seja feito na tentativa de evitar este acontecimento, deste modo é possível concluir que o uso do algoritmo de logistic regression ficou à quem das expetativas.



(a) Averaged Perceptron

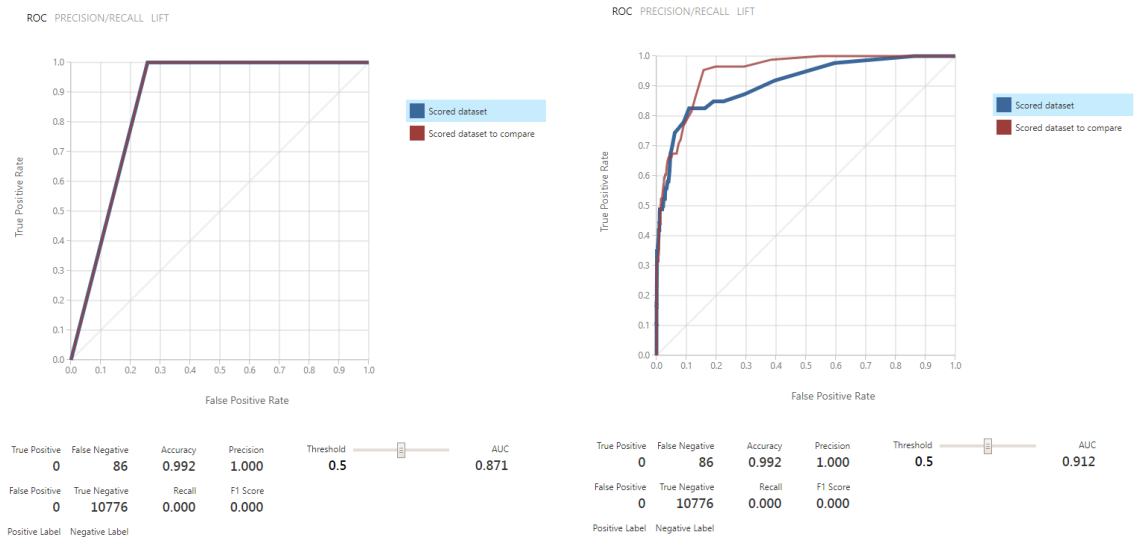


(b) Logistic Regression

Figura 7.16: Métricas de desempenho dos algoritmos de Averaged Perceptron e Logistic Regression para a agregação de cliente, produto e serviço, pela métrica ProcessedDateTime

No que toca aos algoritmos presentes na Figura 7.17 é possível reparar que ambas as curvas (considerar as curvas a azul) se assemelham às curvas apresentadas pelos mesmos algoritmos para a agregação que leva em conta a coluna de ProcessedDateTime mas sem a informação relativa ao serviço especifico. É visível que a falta de entidades classificadas com o valor de 1 está presente em ambos os algoritmos, pelo que a utilização de qualquer um destes algoritmos para implementação do modelo levaria a uma solução que não iria cumprir com o objetivo do projeto, assim sendo, podemos excluir da possibilidade de utilização para construir a solução final tais algoritmos.

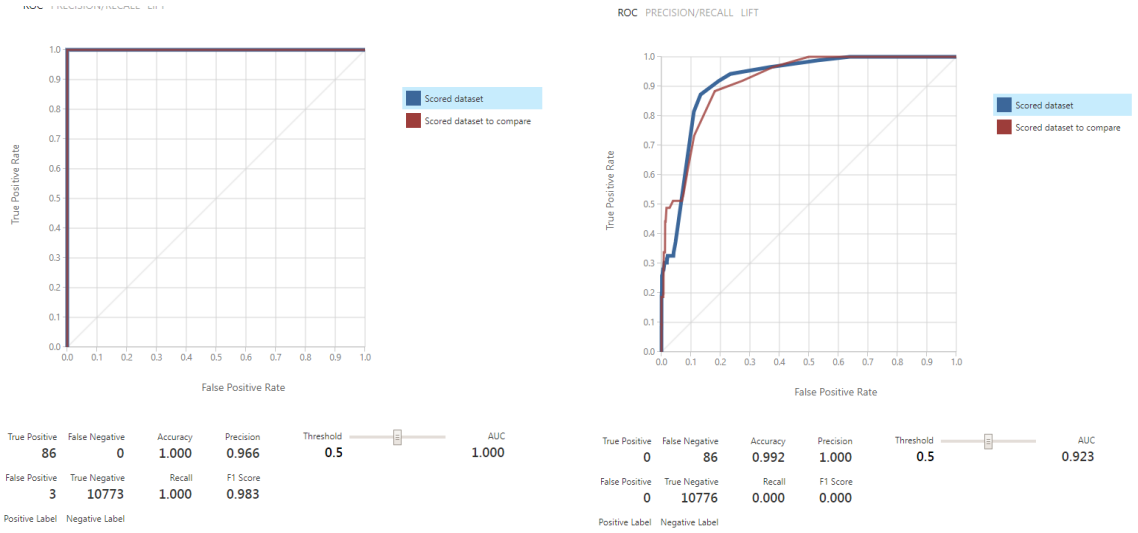
Por fim no que diz respeito aos dois últimos algoritmos, cujos resultados são apresentados na Figura 7.18, é possível verificar que o algoritmo de boosted decision tree na Figura 7.18(a) teve um melhor desempenho quando comprado com o algoritmo de decision jungle na Figura 7.18(b), de notar a falta de entidades classificadas com falsos negativos e o baixo número de casos classificados como falsos positivos, tendo em conta



(a) Bayes Point Machine (b) Decision Florest

Figura 7.17: Métricas de desempenho dos algoritmos de Bayes Point Machine e Decision Florest para a agregação de cliente, produto e serviço, pela métrica ProcessedDateTime

o panorama geral que tem sido verificado para esta agregação de dados em especifico este algoritmo apresenta assim os melhores resultados, contudo este mesmo algoritmo tem prestações melhores quando utilizado em conjunto com outras agregações de dados. Relativamente ao algoritmo de decision jungle este tem problemas semelhantes a alguns dos algoritmos já mencionados, pelo facto de não possuir atribuição da classe de risco de *churn* a nenhuma entidade, assim sendo, pelo mesmo motivo este algoritmo é automaticamente excluído das possibilidades a utilizar para implementar a solução.



(a) Boosted Decision Tree (b) Decision Jungle

Figura 7.18: Métricas de desempenho dos algoritmos de Boosted Decision Tree e Decision Jungle para a agregação de cliente, produto e serviço, pela métrica ProcessedDateTime

De forma geral os resultados obtidos nos testes com as duas agregações que fazem uso da coluna de ProcessedDateTime relatam piores resultados do que aqueles obtidos com a coluna Date, isto poderá indicar que a informação contida na coluna ProcessedDateTime não é tão útil à resolução do problema em questão como a da coluna Date. Uma vez testadas todas as combinações de algoritmos e conjuntos de dados para os dados de subscrições para a previsão de casos de *churn* passa a ser possível tomar uma decisão de qual a melhor combinação a utilizar para dar resposta ao problema, tal decisão é tomada na Secção 7.3, onde todos os fatores são levados em conta e devidamente explicados.

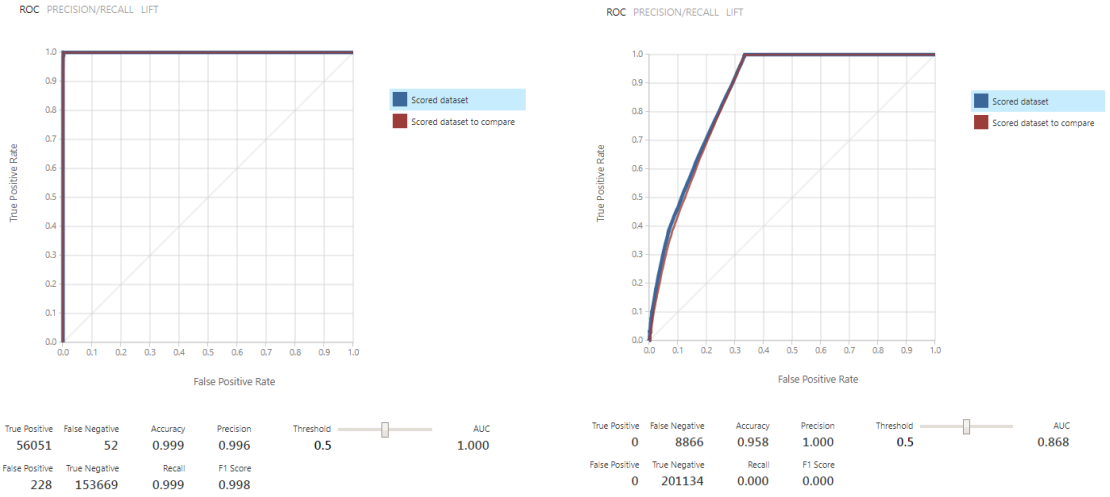
7.1.2 Teste de algoritmos para classificação de upgrades e downgrades

De forma a avaliar os diferentes algoritmos no que toca à deteção de casos de *upgrades* e *downgrades* de pacotes foi necessário adicionar dados ao conjunto dos dados a utilizar. A Microsoft possui uma matriz de ofertas de todas as suas subscrições dos diferentes serviços, nesta matriz existe a indicação relativa aos possíveis *upgrades* para o qual um pacote de subscrição pode ser aumentado, foi com o uso desta matriz que passou a ser possível perceber como os pacotes se relacionam entre si e quando existe uma mudança dentro do mesmo pacote de serviços. De forma análoga foi necessário criar uma matriz de *downgrades*, onde constam as informações sobre todos os pacotes e para que pacotes estes podem alterar em caso de *downgrade*. É com base nestas informações que passa a ser possível identificar casos de alteração de pacotes e utilizar essa informação como etiqueta para que os algoritmos de aprendizagem automática aprendam em função dos dados.

De maneira a realizar um estudo mais conclusivo sobre a prestação dos diferentes algoritmos na previsão de alterações de pacotes nos conjuntos de dados, a avaliação das métricas dos modelos será feita tendo em conta a prestação do algoritmo tanto para os casos de *upgrade* como para os casos de *downgrade*, sendo que para isto será apresentado o resultado da avaliação com os gráficos de imagem lado a lado para os dois casos em estudo. Posto isto segue então a análise dos diferentes algoritmos utilizados em função das diferentes agregações de dados.

7.1.2.1 Agregação por cliente em função do tempo

Para a avaliação do algoritmo de averaged perceptron é de considerar as linhas a azul em ambos os gráficos na Figura 7.19, o resultado obtido para a previsão de *upgrades* é bastante diferente daquele que foi obtido para os *downgrades*, no gráfico apresentado na na Figura 7.19(a) foi possível conseguir uma previsão quase perfeita, apenas com um número remanescente de falsos positivos e falsos negativos, relativamente aos resultados do gráfico apresentado na na Figura 7.19(b) que se refere à tarefa de previsão dos casos em que efetivamente existiu um *downgrade*, como podemos ver pela ausência de verdadeiros positivos e falsos negativos nenhuma entidade viu atribuído a si mesmo o valor de 1, indicando assim que o algoritmo não é o adequado para fazer a previsão de casos de *downgrade* mas é um excelente candidato para *upgrades*.



(a) Situação de Upgrade

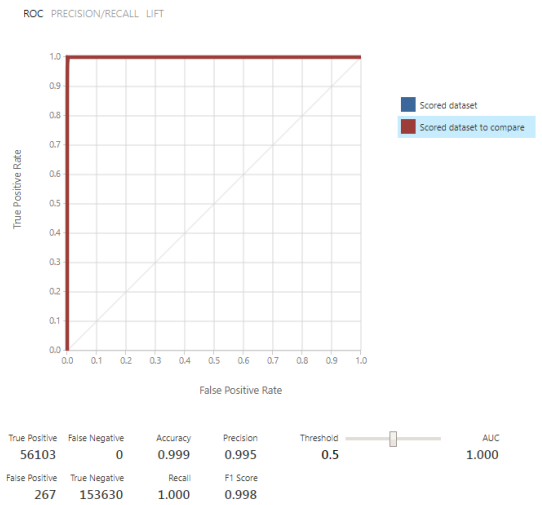
(b) Situação de Downgrade

Figura 7.19: Métricas de desempenho do algoritmo de Averaged Perceptron para *upgrade* e *downgrade* de pacote, agregação por cliente

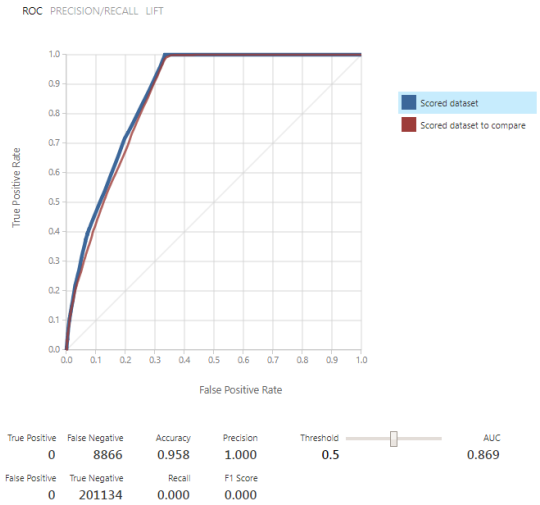
Apesar do algoritmo de averaged perceptron apresentar valor bastante adequados para a classificação de situações de *upgrade*, o algoritmo de logistic regression conseguiu ter valores sensivelmente melhores, como é possível verificar na Figura 7.20. A ausência de entidades classificadas como falso negativo revela que com a utilização deste algoritmo é possível não deixar passar nenhum tipo de oportunidade de negócio, uma vez que o algoritmo conseguiu detetar todas as potenciais situações de aumento do pacote da subscrição. Assim sendo, este fato torna o algoritmo de logistic regression um melhor candidato à implementação da solução final. Relativamente à previsão de casos de *downgrade* também este algoritmo teve uma prestação abaixo do expectável, uma vez que não identificou nenhuma entidade para efetuar *downgrade*.

O algoritmo de bayes point machine trouxe algumas mudanças relativas ao resultado do problema da previsão, como é visivel na Figura 7.21, este algoritmo foi o primeiro a conseguir classificar corretamente entidades que realizaram um *downgrade* ao seu serviço de subscrição, apesar do número total de entidades classificadas como positivas ser bastante baixo, a existência desta classificação abre a possibilidade de ser melhorada com a utilização de mais dados ao longo do tempo. Um facto denotado foi a existência de um baixo número de *downgrades* no sistema, os dados reais extraídos dos clientes efetivamente apresentam uma baixa percentagem de *downgrades*, esta baixa amostragem tem um impacto direto nos classificadores, uma vez existem poucos dados sobre os quais extrair informação relativa à forma como as métricas influenciam o resultado final. No que toca aos resultados para os casos de *upgrades* este foram exatamente iguais aos apresentados pelo algoritmo de logistic regression, apresentando por isso uma excelente prestação.

O teste com o algoritmo de decision forest não trouxe resultados diferentes para o estudo do impacto da classificação de entidades, tanto no que toca à previsão de casos

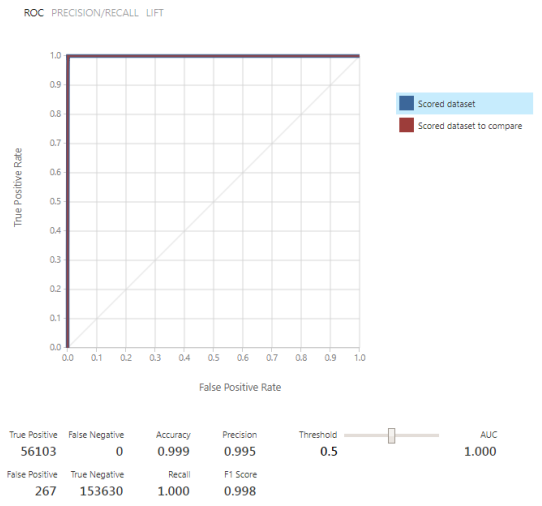


(a) Situação de Upgrade

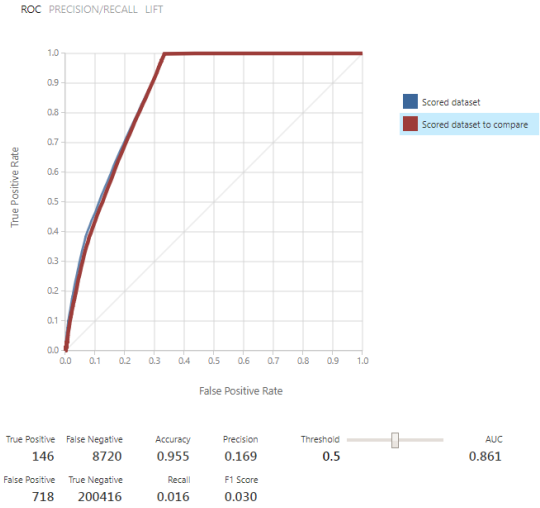


(b) Situação de Downgrade

Figura 7.20: Métricas de desempenho do algoritmo de Logistic Regression para *upgrade* e *downgrade* de pacote, agregação por cliente



(a) Situação de Upgrade

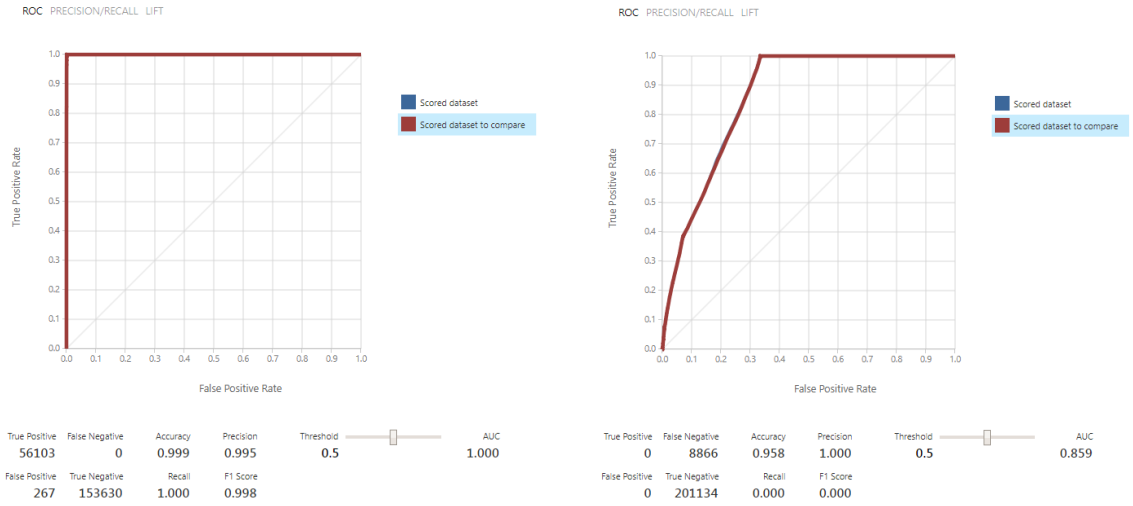


(b) Situação de Downgrade

Figura 7.21: Métricas de desempenho do algoritmo de Bayes Point Machine para *upgrade* e *downgrade* de pacote, agregação por cliente

de *upgrade* como no que toca à previsão de casos de *upgrades* os resultados foram iguais aos obtidos pelos algoritmos de bayes point machine e logistic regression, o que nesta altura começou a suscitar algumas dúvidas relativas a se efetivamente existiam 267 entidades cujas características fossem diferentes daquilo que o algoritmo assumiria para a classificação como *update* e por isso é que existia sempre este exato número fixo de falsos positivos.

Com a obtenção dos resultados para o algoritmo de boosted decision tree confirmou-se que afinal não existem 267 entidades com características diferentes, uma vez que

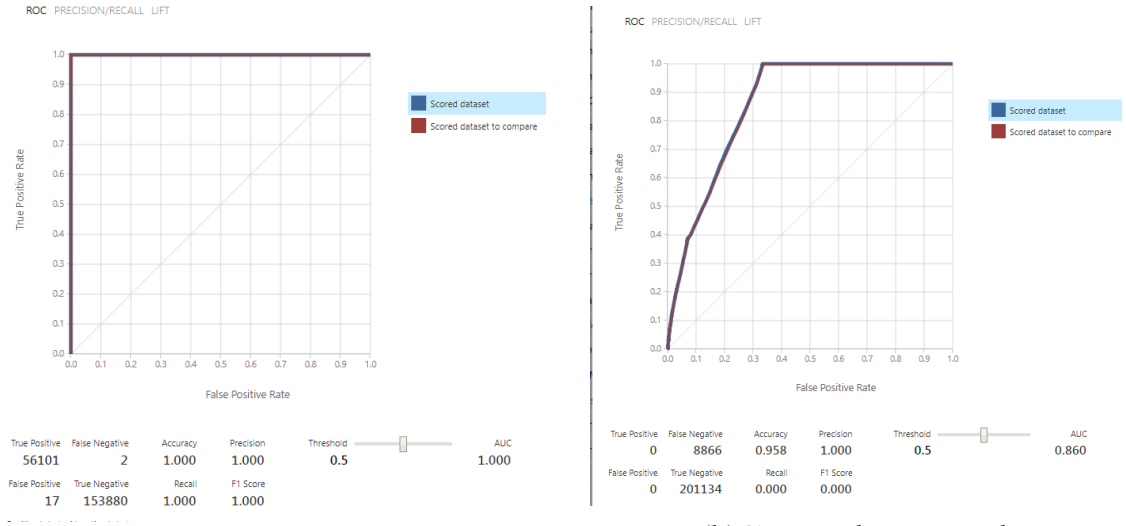


(a) Situação de Upgrade

(b) Situação de Downgrade

Figura 7.22: Métricas de desempenho do algoritmo de Decision Forest para *upgrade* e *downgrade* de pacote, agregação por cliente

o algoritmo apenas classificou como falsos positivos 17 entidades, como é visível na Figura 7.23. Este algoritmo, apesar de apresentar um total de 2 casos de falsos negativo e isso ser efetivamente uma limitação do algoritmo, permite obter métricas de avaliação perfeitas, fazendo assim com que a *accuracy*, *precision*, *recall e f1 score* tenham o valor de 1. Infelizmente no que toca à análise do gráfico em (b) não existiu qualquer mudança, sendo que até este momento apenas o algoritmo de bayes point machine conseguiu classificar positivamente entidades do conjunto de dados.

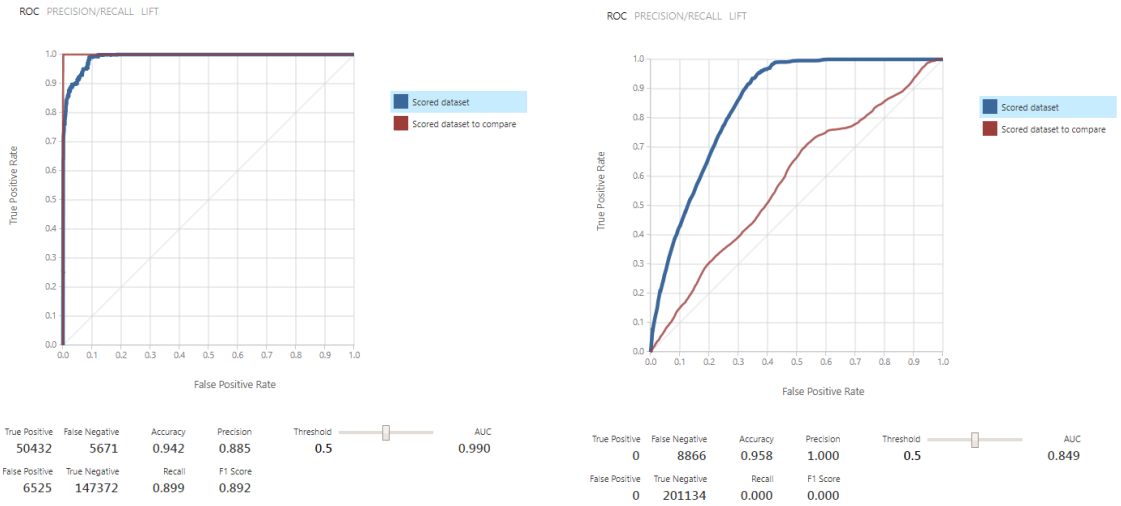


(a) Situação de Upgrade

(b) Situação de Downgrade

Figura 7.23: Métricas de desempenho do algoritmo de Boosted Decision Tree para *upgrade* e *downgrade* de pacote, agregação por cliente

O teste relativo ao último algoritmo para esta agregação de dados conduziu a resultados um pouco diferentes, até agora todos os testes de previsão para dados de *upgrades* obtiveram curvas perfeitas com a presença de uma pequena quantidade de falsos positivos e falsos negativos, a utilização do algoritmo de decision jungle para a formação do modelo de previsão conduziu à existência de uma pequena quebra na curva perfeita para dados elevados de positivos quando o rácio de negativos é baixo, como pode ser observado visualmente na Figura 7.24, esta particularidade fez com que existisse um aumento de falsos positivos e falsos negativos. Devido ao facto de existirem algoritmos com resultados muito melhores a nível de desempenho, os resultados obtidos pelas métricas de avaliação deste algoritmo fazem com que a sua utilização na solução final seja rejeitada pelo facto de existiram melhores candidatos.



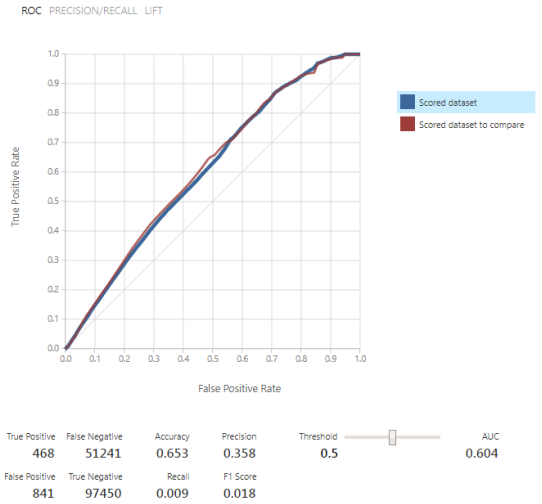
(a) Situação de Upgrade

(b) Situação de Downgrade

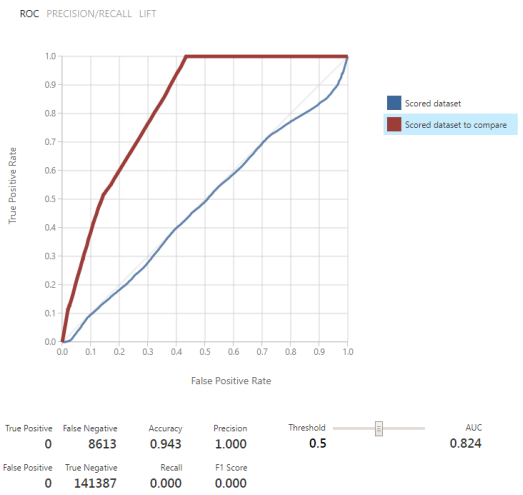
Figura 7.24: Métricas de desempenho do algoritmo de Decision Jungle para *upgrade* e *downgrade* de pacote, agregação por cliente

7.1.2.2 Agregação por cliente e subscrição em função do tempo

Relativamente ao algoritmo de average perceptron é de considerar a linha azul no gráfico da esquerda e a linha a vermelho no gráfico da direita na Figura 7.25. Apesar da AUC do gráfico relativo aos *downgrades* mostrar um valor consideravelmente maior do que o gráfico de *upgrades* este algoritmo mostrou uma performance inferior como é possível verificar pela métrica de *Recall* e *F1 Score*, o facto do algoritmo não ter conseguido detetar situações de *downgrade* torna-se uma limitação no que diz respeito ao uso do mesmo para utilização futura na implementação da solução. Os resultados obtidos por parte deste algoritmo ficaram muito aquém dos obtidos pelo mesmo quando utilizada a agregação de dados apenas por cliente no caso dos casos de *upgrades*, o que pode ser um indicador de que esta agregação de dados irá produzir resultados inferiores, contudo é necessário testar com todos os algoritmos pois podem existir surpresas que indiquem o contrário.



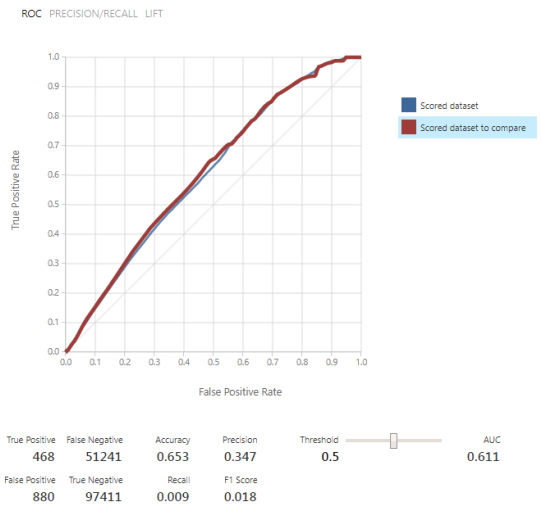
(a) Situação de Upgrade



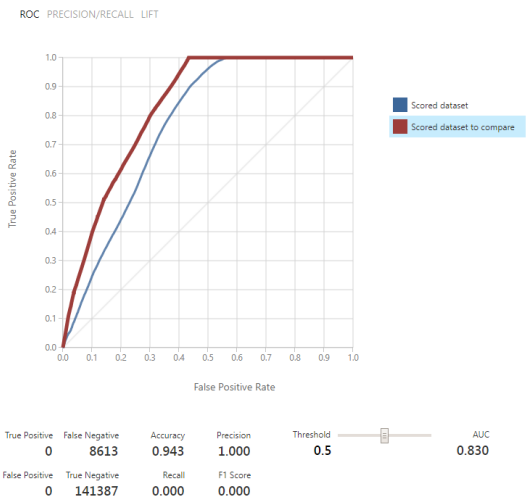
(b) Situação de Downgrade

Figura 7.25: Métricas de desempenho do algoritmo de Averaged Perceptron para *upgrade* e *downgrade* de pacotes, agregação por cliente e subscrição

Com a utilização do algoritmo de logistic regression começou a detetar-se um padrão nos resultados, pelo menos para os dados de agregação apenas por cliente, a deteção de casos de *upgrade* com este algoritmo (Figura 7.26) teve uma prestação bastante similar à da obtida com a utilização do algoritmo de averaged perceptron, sendo esta uma previsão que deteta efetivamente a existência de casos onde se verificou a mudança de pacote mas apresenta ao mesmo tempo um elevado número de falsos negativos, deixado assim passar muitas oportunidades de negócio. Relativamente à previsão de situações de *downgrade*, representada a vermelho no gráfico da direita, esta demonstrou que mais uma vez foi impossível para o algoritmo detetar situações de descida de pacote de subscrição.



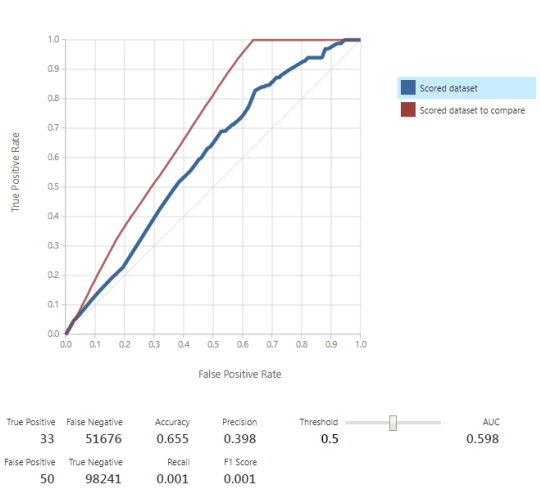
(a) Situação de Upgrade



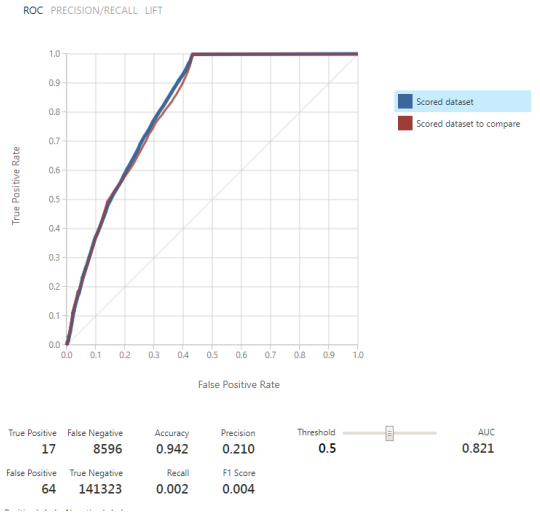
(b) Situação de Downgrade

Figura 7.26: Métricas de desempenho do algoritmo de Logistic Regression para *upgrade* e *downgrade* de pacotes, agregação por cliente e subscrição

O algoritmo de Bayes Point Machine obteve uma prestação um pouco abaixo dos algoritmos apresentados até este ponto. Como é possível verificar nos dois gráficos (linha azul) da Figura 7.27, na situação de previsão de *upgrades*, apesar de métricas como a *accuracy* e *precision* apresentarem valores semelhantes aos dos algoritmos passados, os valores de *recall* e *F1 score* são quase nulos, isto deve-se à grande diminuição de verdadeiros positivos obtidos pelo modelo, esta classificação coloca o algoritmo de bayes point machine atrás das restantes alternativas, devido ao seu desempenho. Por outro lado este algoritmo foi o primeiro a conseguir classificar entidades corretamente no que toca à deteção de casos de *upgrade*, contudo, a fraca prestação apresentada pelo modelo não garante confiança suficiente para que este seja utilizado para tal previsão, uma vez que apenas foram detetadas 17 entidades num universo de mais de 8000.



(a) Situação de Upgrade

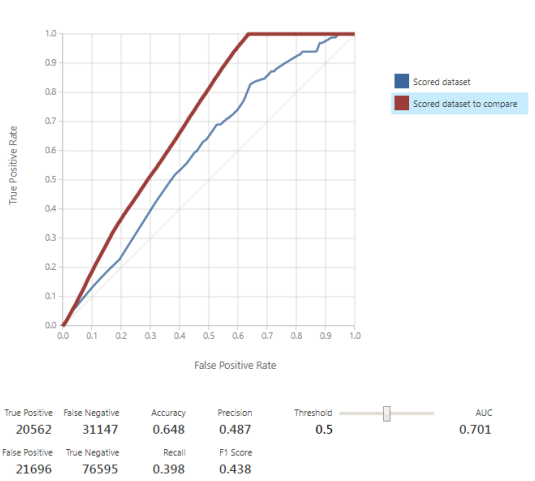


(b) Situação de Downgrade

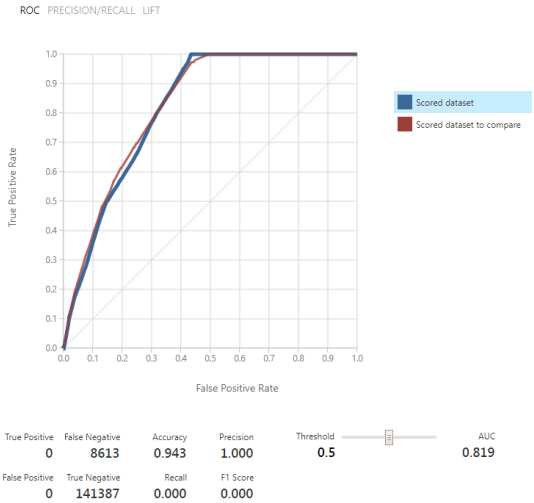
Figura 7.27: Métricas de desempenho do algoritmo de Bayes Point Machine para *upgrade* e *downgrade* de pacotes, agregação por cliente e subscrição

O algoritmo de Decision Forest vem trazer um desempenho superior a todos os algoritmos reportados no documento até agora para esta agregação de dados. A classificação de casos de *upgrade*, verificada a linha vermelha da Figura 7.28(a), apresenta uma elevada deteção de casos positivos, que até agora não havia sido alcançada com o uso de outros algoritmos, esta subida levou também à grande mudança nos valores de *recall* e *F1 score*, aumentando assim a classificação geral do algoritmo no que toca à sua utilização para a solução. Como tem vindo a ser uma constante nestes testes o caso da deteção de situações de *downgrades* não tem apresentado os melhores resultados, sendo que mais vez não foi possível detetar a presença de nenhuma situação de *downgrade*.

O algoritmo de Boosted Decision Tree, cujos resultados se encontram reportados na Figura 7.29, obteve resultados bastante semelhantes ao algoritmo de Decision Forest, sendo que ainda assim conseguiu alcançar valores de métricas ligeiramente superior. A não deteção de situações de *downgrade* continua a ser uma constante para todos os



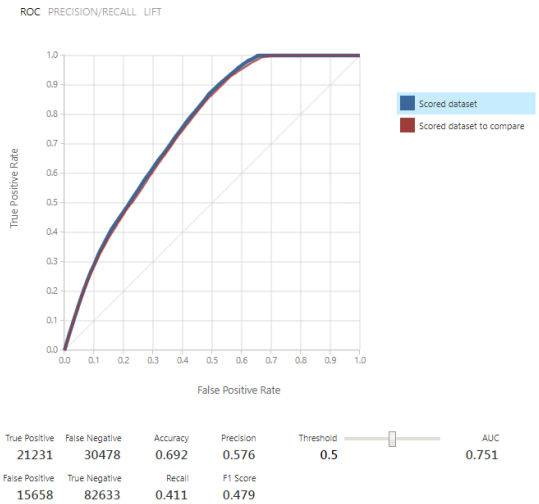
(a) Situação de Upgrade



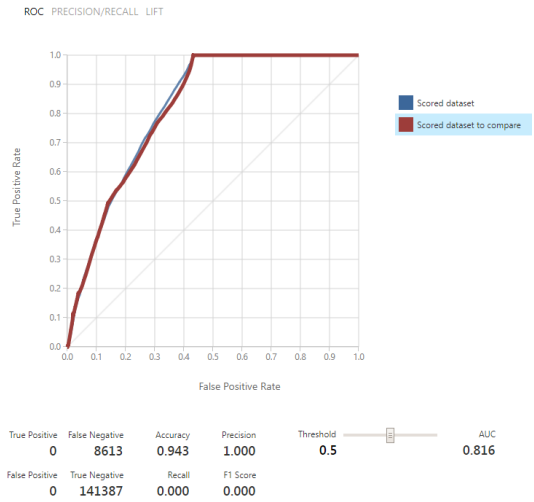
(b) Situação de Downgrade

Figura 7.28: Métricas de desempenho do algoritmo de Decision Forest para *upgrade* e *downgrade* de pacotes, agregação por cliente e subscrição

algoritmos em análise, sendo que a partir deste momento esta deixou de ser levada em conta para cenários de desempate, uma vez que os resultados obtidos pelos modelos para este problema têm vindo a indicar o mesmo no caso especifico dos *downgrades*.



(a) Situação de Upgrade

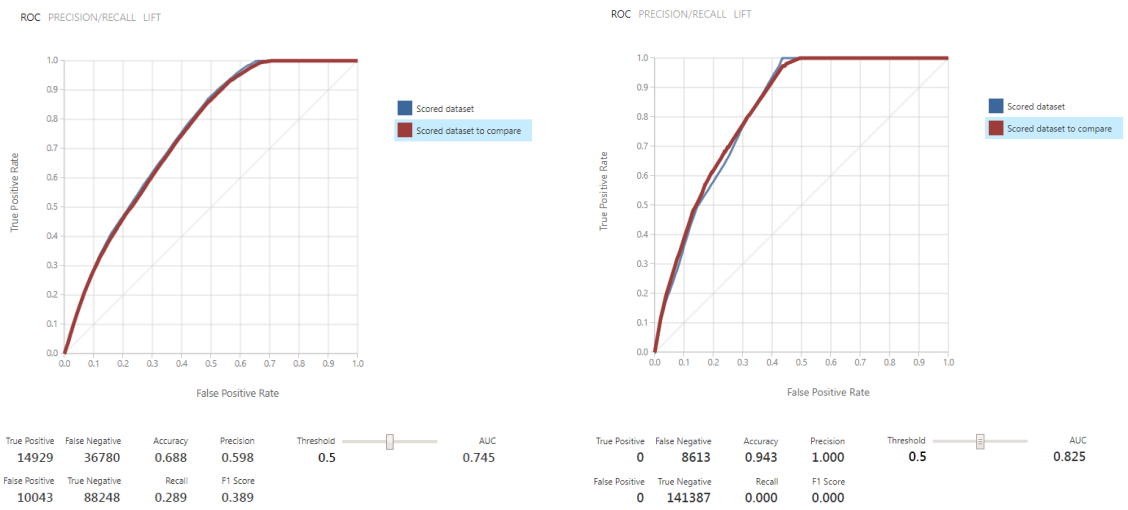


(b) Situação de Downgrade

Figura 7.29: Métricas de desempenho do algoritmo de Boosted Decision Tree para *upgrade* e *downgrade* de pacotes, agregação por cliente e subscrição

A utilização de algoritmos baseados em árvores de decisão, sem dúvida, apresenta um bom desempenho no que toca á identificação de casos de *upgrades*, tanto o algoritmo de decision forest, boosted decision tree e agora o algoritmo de decision jungle, cujos resultados se encontram representados na Figura 7.30, apresentaram valores significativamente

superiores aos valores conseguidos pelos restantes algoritmos quando aplicados ao conjunto de dados em questão. A própria estrutura dos dados potencia este resultado, pois ao estarmos a treinar modelos para detetar situações de alteração de pacotes, a própria estrutura que existe na hierarquia de mudanças de pacote Microsoft por si só já tem o aspeto de uma árvore. Assim sendo, encerramos a análise da prestação dos diferentes modelos para a agregação de cliente e suas subscrições em função do tempo com a conclusão de que os algoritmos que têm por base a utilização de árvores de decisão têm melhor desempenho para deteção de casos de *upgrade*.



(a) Situação de Upgrade

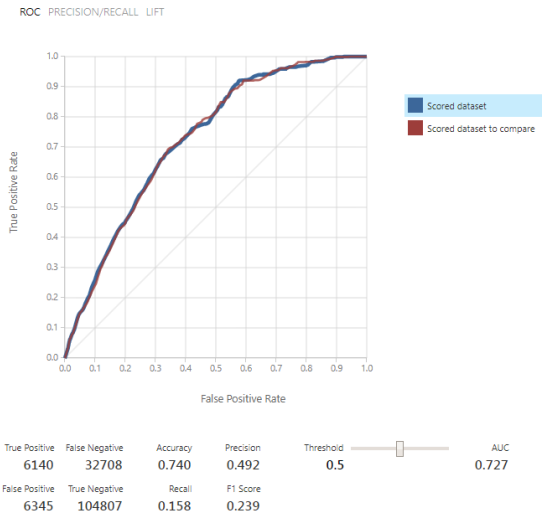
(b) Situação de Downgrade

Figura 7.30: Métricas de desempenho do algoritmo de Decision Jungle para *upgrade* e *downgrade* de pacotes, agregação por cliente e subscrição

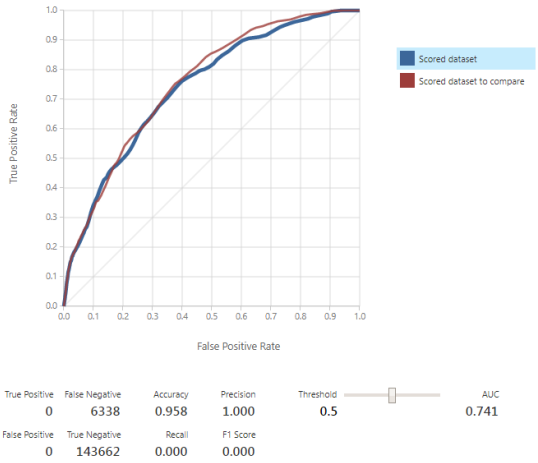
7.1.2.3 Agregação pela métrica Date, Cliente + Produto

Passando agora à análise relativa à agregação que tem em conta o cliente e o produto ativado pela 'Date' começamos a análise uma vez mais pelo algoritmo de averaged perceptron, cujos resultados se encontram expressos na Figura 7.31. Comparativamente com a agregação anteriormente analisada, no que toca à previsão de *upgrades* o algoritmo apresentou resultados consideravelmente melhores, o número de verdadeiros positivos subiu e conta com cerca de menos vinte mil ocorrências de falsos negativos, o que do ponto de vista do negócio é excelente, contudo esta prestação revela-se um pouco débil para ser aplicada como solução final ao problema, como podemos verificar pelos baixos valores de *recall* e *F1 score*. Mais uma vez não foi possível estabelecer uma relação entre os dados e a classe correspondente no que toca à previsão de casos de *downgrade*, pelo que o algoritmo classificou todas as entidades com a mesma classe.

Relativamente ao algoritmo de logistic regression este produziu resultados bastante semelhantes aos obtidos pelo algoritmo de averaged perceptron, quer para o caso da previsão de *upgrades* como de *downgrades*. De facto ambos os algoritmos têm vindo a



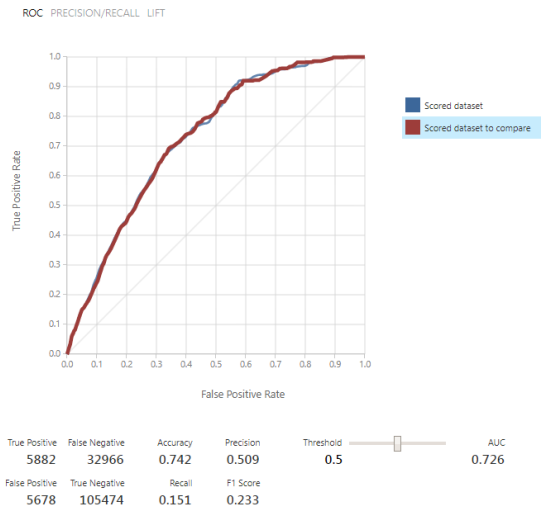
(a) Situação de Upgrade



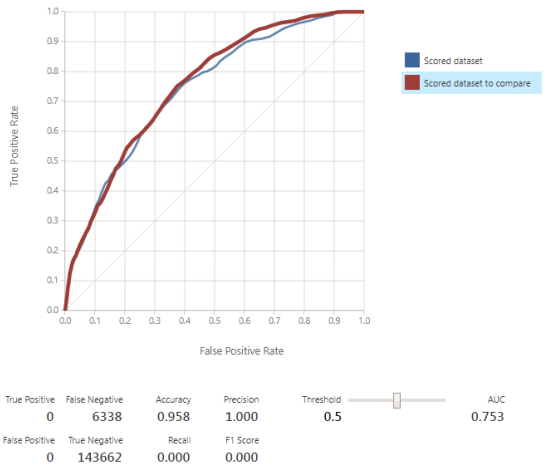
(b) Situação de Downgrade

Figura 7.31: Métricas de desempenho do algoritmo de Averaged Perceptron para *upgrade* e *downgrade* de pacotes, agregação dos clientes e produtos tendo em conta a coluna Date

demonstrar que no contexto dos dados utilizados neste caso de estudo, estes tendem a ter as mesmas dificuldades em estabelecer as relações entre os dados de acordo com o objetivo de classificação, assim sendo acabam por produzir resultados bastante próximos. Podemos observar os resultados do teste realizado a este algoritmo na Figura 7.32. A linha a tomar em consideração para análise do algoritmo é a linha traçada a vermelho, sendo que a linha traçada a azul representa o algoritmo de averaged perceptron, aqui mais uma vez podemos ver como existem poucas diferenças entre o resultado final da classificação com os dois algoritmos.



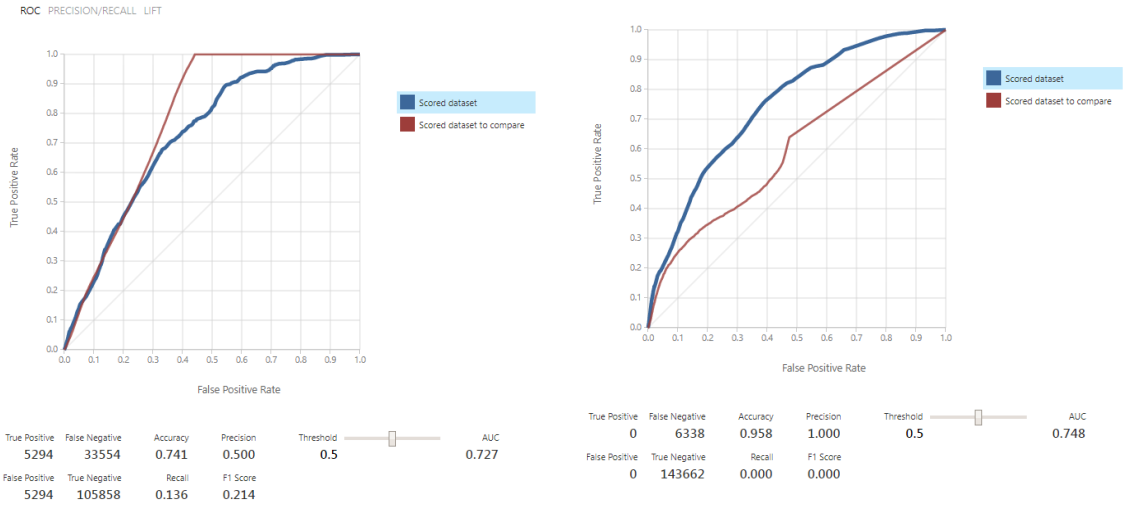
(a) Situação de Upgrade



(b) Situação de Downgrade

Figura 7.32: Métricas de desempenho do algoritmo de LogisticRegression para *upgrade* e *downgrade* de pacotes, agregação dos clientes e produtos tendo em conta a coluna Date

A utilização do algoritmo de bayes point machine para a classificação relativa a situações de *update* revelou-se significativamente melhor do que a obtida pelo mesmo algoritmo com a agregação de clientes e subscrições. Apesar dos resultados apresentados na Figura 7.33 indicarem que esta não foi a melhor prestação do algoritmo até agora, pois na agregação apenas por cliente este apresentou resultados fortemente posítivos, contudo a informação possível de retirar deste teste é de que provavelmente a agregação de cliente e subscrições poderá não ser a agregação com mais valor, contudo qualquer conclusão relativa a algoritmos e agregações será apenas discutida na Secção 7.3.



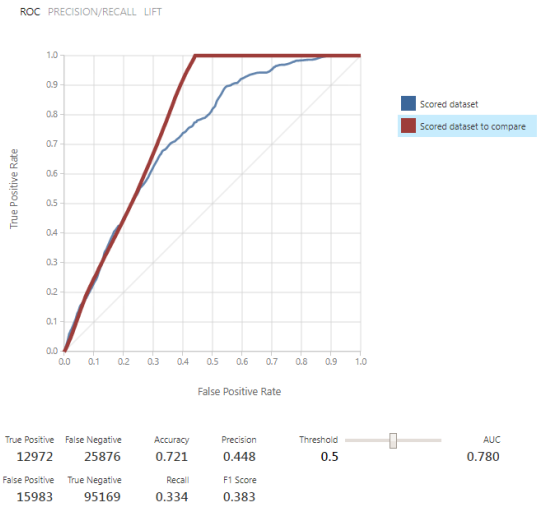
(a) Situação de Upgrade

(b) Situação de Downgrade

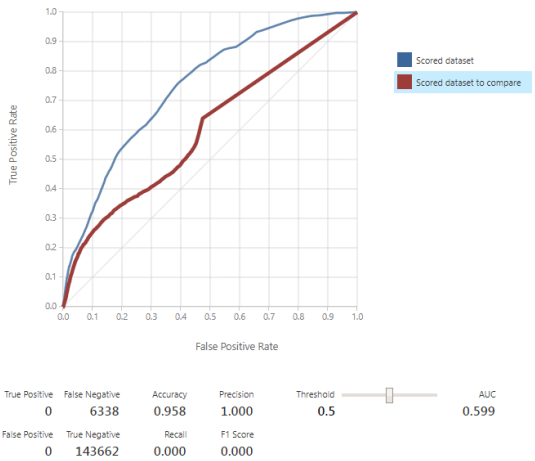
Figura 7.33: Métricas de desempenho do algoritmo de Bayes Point Machine para *upgrade* e *downgrade* de pacotes, agregação dos clientes e produtos tendo em conta a coluna Date

Passando agora para os algoritmos com base em árvores de decisão, nomeadamente o algoritmo de decision forest, este obteve os melhores resultados para o conjunto de dados em análise. Os resultados podem ser consultados com recurso à Figura 7.34, onde é de considerar apenas a linha a vermelho. Foi possível obter resultados medianos no que toca ao gráfico Figura 7.34(a), onde as métricas de análise indicam que o algoritmo conseguiu efetivamente traçar algumas relações entre os dados e as classes a classificar. Embora os resultados não sejam excelentes, estes são algo animadores, pois conseguimos perceber que diferentes agregações conseguem efetivamente fazer com que os algoritmos retirem algumas informações e consigam aprender a partir dos seus dados. O caso da previsão de *downgrades* continua constante, uma vez que os resultados relativos a entidades classificadas positivamente não sofrem alterações.

No caso particular dos últimos dois algoritmos em análise para esta agregação, o desempenho dos mesmos foi similar. Os resultados para o algoritmo de boosted decision tree encontram-se explícitos na Figura 7.35 e no que toca ao algoritmo de decision jungle estes constam na Figura 7.36. De notar que a curva em análise na Figura 7.35 tem uma prestação bastante similar à obtida pelo algoritmo de decision jungle, tanto para situações



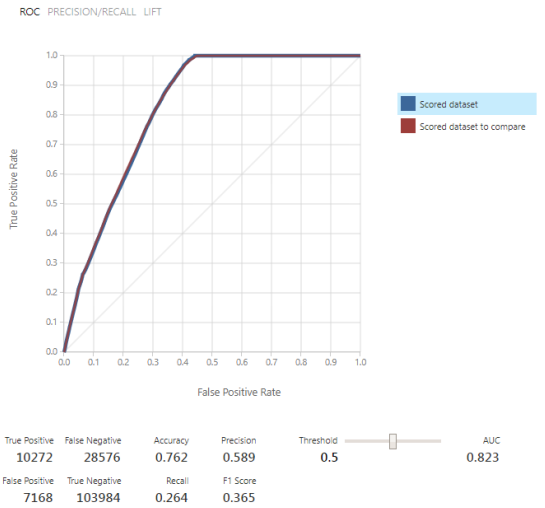
(a) Situação de Upgrade



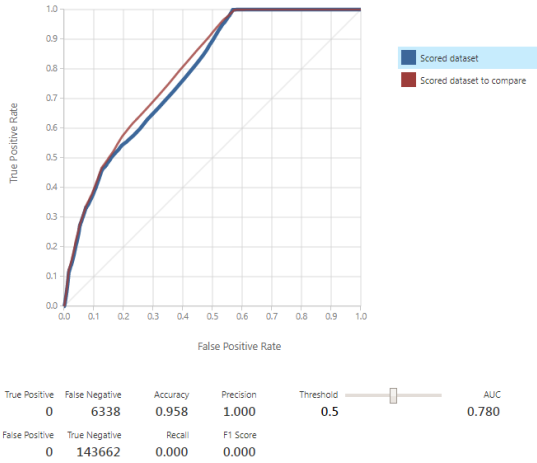
(b) Situação de Downgrade

Figura 7.34: Métricas de desempenho do algoritmo de Decision Forest para *upgrade* e *downgrade* de pacotes, agregação dos clientes e produtos tendo em conta a coluna Date

de *upgrade* como para situações de *downgrade*. A vantagem destes dois algoritmos terem desempenhos e resultados quase iguais é que é possivel automaticamente exclui-los da possibilidade de utilização para implementação da solução, uma vez que este não possuem as melhores métricas de avaliação para a presente agregação de dados, isto permite diminuir as possibilidades de escolha e acelerar o processo de conclusão sobre que modelo utilizar.

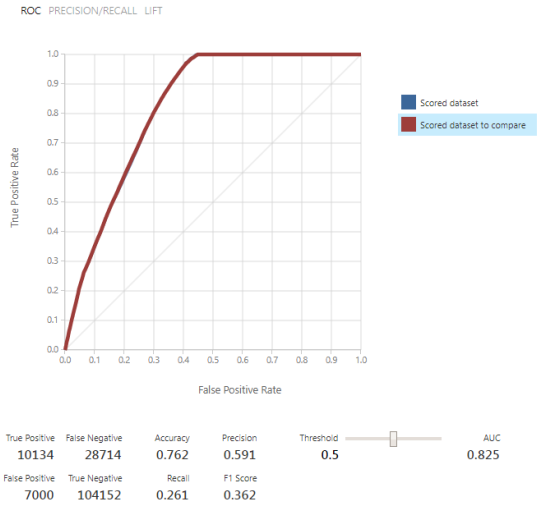


(a) Situação de Upgrade

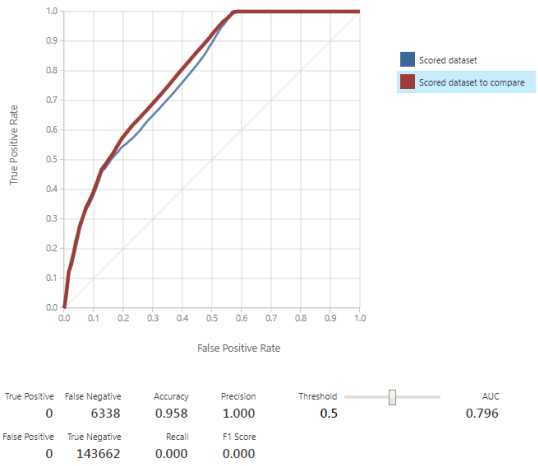


(b) Situação de Downgrade

Figura 7.35: Métricas de desempenho do algoritmo de Boosted Decision Tree para *upgrade* e *downgrade* de pacotes, agregação dos clientes e produtos tendo em conta a coluna Date



(a) Situação de Upgrade



(b) Situação de Downgrade

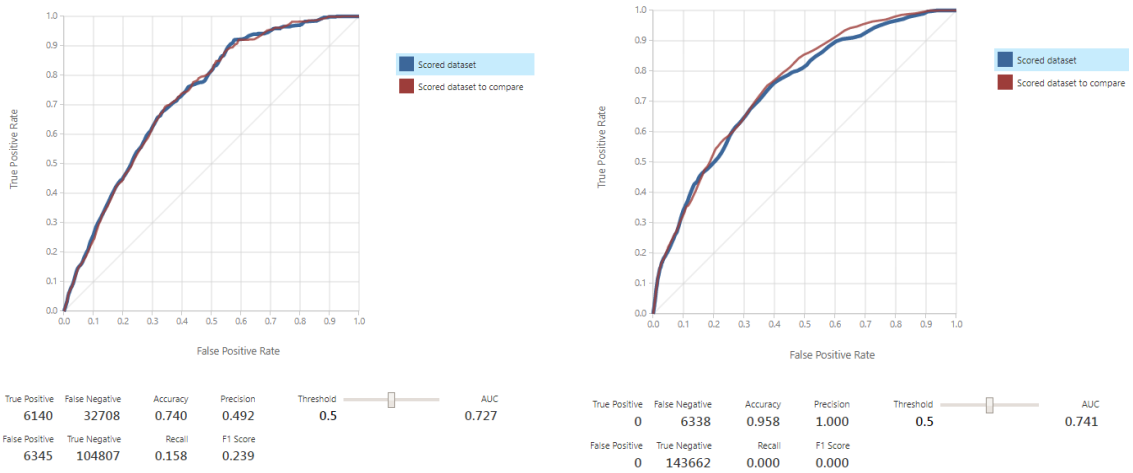
Figura 7.36: Métricas de desempenho do algoritmo de Decision Jungle para *upgrade* e *downgrade* de pacotes, agregação dos clientes e produtos tendo em conta a coluna Date

7.1.2.4 Agregação pela métrica Date, Cliente + Produto + Serviço

A Figura 7.37 reporta os resultados obtidos para o algoritmo de averaged perceptron para a agregação de dados em estudo. Apesar da curva representada nos dois gráficos ser bastante similar os resultados presentes na matriz de confusão indicam que a prestação dos algoritmos não é de todo idêntica. Mais uma vez torna-se evidente que a deteção de casos de *downgrade* não está a acontecer, mesmo alterando o conjunto de dados em estudo. Relativamente à deteção de casos de *upgrades* os valores são um pouco melhores, contudo ficam muito aquém daqueles que já foram obtidos por este algoritmo no contexto do problema, o que poderá ser um indicador de que esta agregação poderá não produzir os melhores resultados.

Mesmo com a alteração da agregação de dados em estudo mais uma vez o algoritmo de logistic regression obteve uma prestação idêntica ao algoritmo de averaged perceptron, tal como é possível confirmar recorrendo à Figura 7.38. Os dados relativos à deteção de situações de potencial *upgrade* são bastante semelhantes, pelos que não é conclusivo qual dos dois algoritmos poderão ter um desempenho realmente superior, pois estas pequenas diferenças podem ver os seus valores oscilados consoante pequenas alterações no conjunto de treino. Para o problema dos casos de *downgrade* mais uma vez os resultados foram não conclusivos no que toca à escolha de um algoritmo capaz de aprender com base dos dados fornecidos.

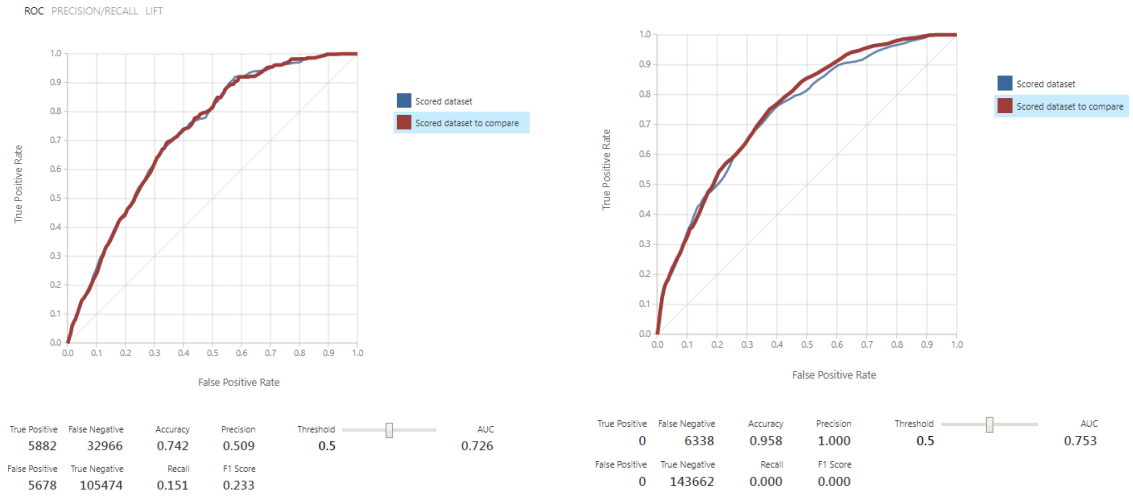
Uma vez que continua a fazer sentido testar os restantes algoritmos de forma a completar a sequência de testes segue-se a análise ao algoritmo de bayes point machine, cujos resultados constam na Figura 7.39. (linhas a azul). Ao contrário do que já aconteceu com o algoritmo com outros conjuntos de dados desta vez o algoritmo de bayes point machine não conseguiu classificar uma única entidade como potencial *downgrade*, apresentando



(a) Situação de Upgrade

(b) Situação de Downgrade

Figura 7.37: Métricas de desempenho do algoritmo de Averaged Perceptron para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica Date



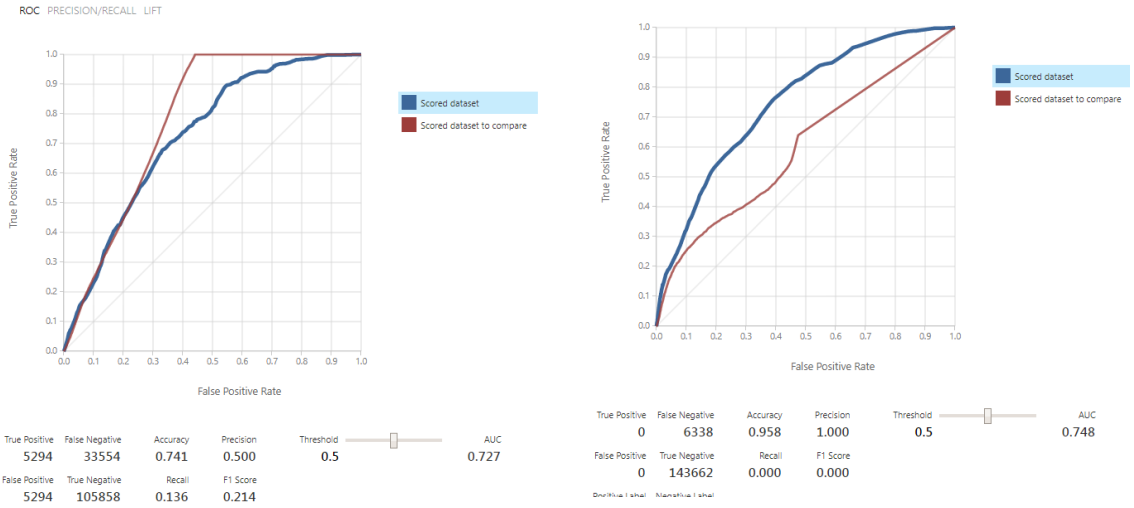
(a) Situação de Upgrade

(b) Situação de Downgrade

Figura 7.38: Métricas de desempenho do algoritmo de Logistic Regression para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica Date

assim um resultado equivalente a todos os outros para a deteção de casos de redução de pacote. Para o problema da previsão de *upgrades* os resultados não foram tão bons como no caso de average perceptron ou de logistic regression, pelo que se for concluído que se deve utilizar esta agregação de dados, esta não será utilizada em conjunto com o algoritmo de bayes point machine.

Os resultados do algoritmo de decision forest, presentes em vermelho na Figura 7.40, apresentam resultados bastante superiores do ponto de vista de *recall* e de *F1 score* do que todos os algoritmos utilizados até agora para esta agregação de dados, embora não sendo um exemplo de um classificador excelente, a sua prestação merece alguma atenção

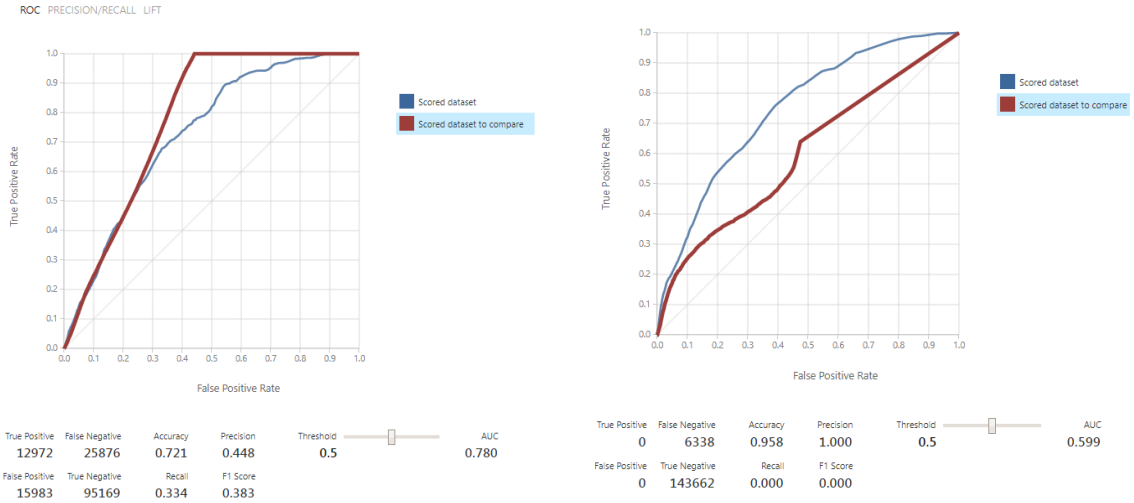


(a) Situação de Upgrade

(b) Situação de Downgrade

Figura 7.39: Métricas de desempenho do algoritmo de Bayes Point Machine para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica Date

no caso da deteção de casos de *upgrades*.



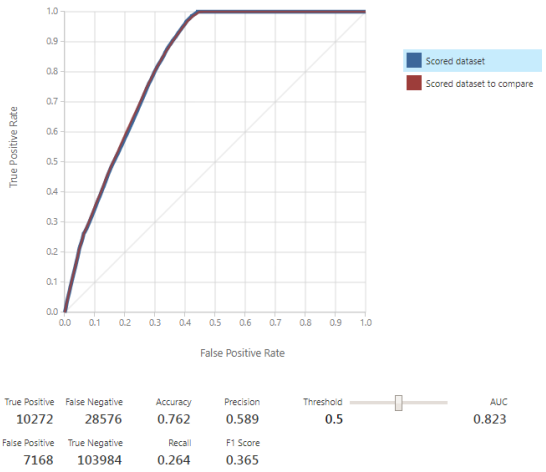
(a) Situação de Upgrade

(b) Situação de Downgrade

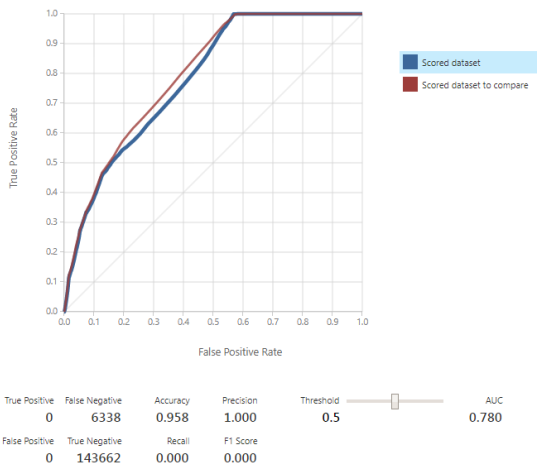
Figura 7.40: Métricas de desempenho do algoritmo de Decision Forest para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica Date

A análise para os algoritmos de boosted decision tree e de decision jungle poderá ser realizada em conjunto, ambos os algoritmos apresentam curvas de aprendizagem idênticas, com resultados para as métricas em análise semelhantes. O resultado do teste do algoritmo de boosted decision tree encontra-se representado a azul na Figura 7.41 e o resultado para o algoritmo de decision jungle encontra-se representado a vermelho na Figura 7.42. As curvas de aprendizagem têm uma curvatura semelhante à adquirida para o modelo treinado com o algoritmo de decision forest, isto faz sentido uma vez que os

três algoritmos têm por base o uso de árvores de decisão para efetuar a sua classificação, contudo, como no caso do boosted decision tree e no caso da decision forest os classificadores utilizam diferentes árvores para adquirir a sua pontuação é de notar uma curvatura menos abrupta na curva de aprendizagem, resultante dos valores médios obtidos ao longo da classificação.

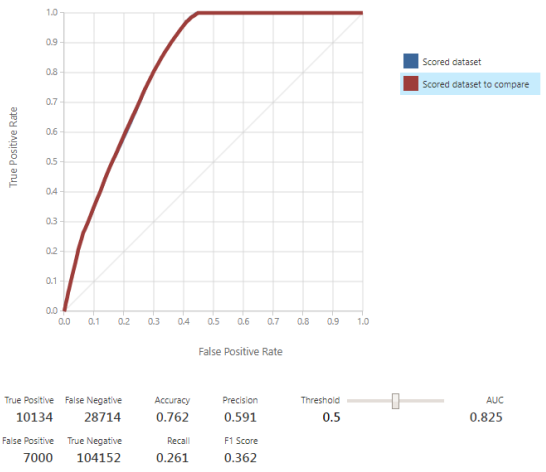


(a) Situação de Upgrade

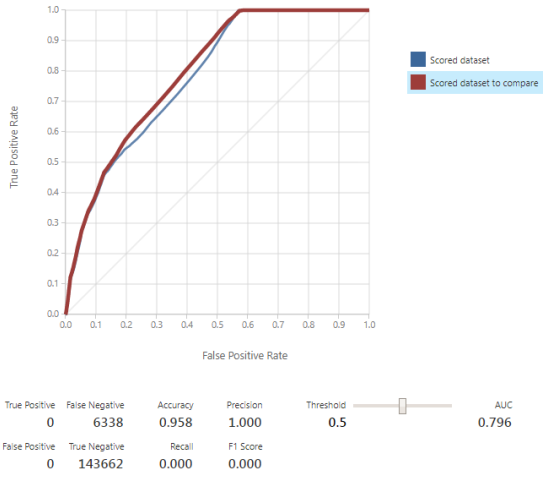


(b) Situação de Downgrade

Figura 7.41: Métricas de desempenho do algoritmo de Boosted Decision Tree para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica Date



(a) Situação de Upgrade



(b) Situação de Downgrade

Figura 7.42: Métricas de desempenho do algoritmo de Decision Jungle para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica Date

Com os testes finalizados para a agregação que tem em conta a coluna da Date para realizar o agrupamento de dados segundo o cliente, produto e serviço utilizado podemos concluir que nenhum dos classificadores obteve um desempenho completamente positivo no que toca à deteção de casos de *downgrade* de pacotes, uma vez que daquilo que

observamos nos dados não existiu uma única entidade que visse atribuída a si mesma o valor positivo de 1. Resta por isso testar se a agregação que tem em conta a coluna de ProcessedDateTime tem ou não melhores resultados para o problema da deteção de subidas ou descidas nos pacotes de produtos dos clientes.

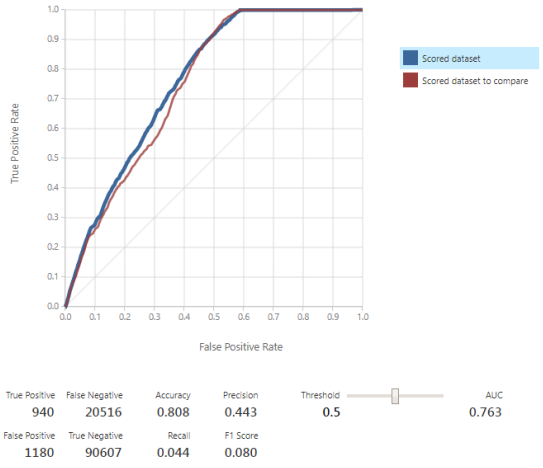
7.1.2.5 Agregação pela métrica ProcessedDateTime, Cliente + Produto

As agregações realizadas tendo em conta a coluna ProcessedDateTime são agregações que possuem uma menor quantidade de entradas, pelo facto da própria coluna ProcessedDateTime possuir uma quantidade de valores únicos menor, como tal, é expectável que os algoritmos de aprendizagem automática apresentem um desempenho um pouco inferior quando comparados com os mesmo algoritmos para os conjuntos de dados que tiveram em vista a informação da coluna Date, isto porque quanto falamos de aprendizagem automática os algoritmos aprendem tendo em conta a informação disponível, sendo que quanto maior for a base de conhecimento, então existem mais casos possíveis de cobrir e consequentemente, melhores serão os resultados dos classificadores. Com isto em mente prosseguiu-se com a análise dos diferentes algoritmos para a agregação de dados realizada.

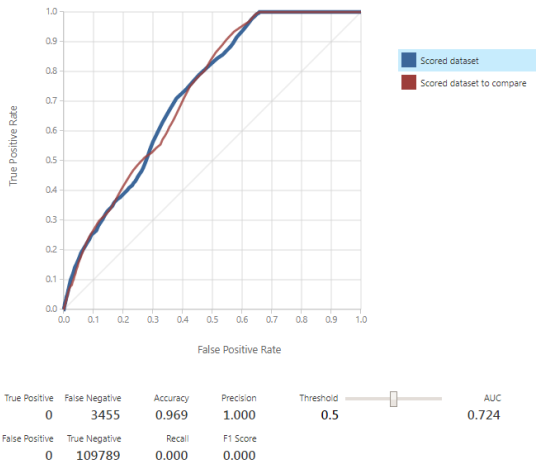
Como seria de esperar os resultados relativos à utilização do algoritmo de averaged perceptron diminuíram quando este foi aplicado ao conjunto de dados que tem por base a coluna ProcessedDateTime em vez da coluna Date. Os resultados deste teste encontram-se apresentados na Figura 7.43. Como é possível confirmar ocorreu uma quebra significativa no número de verdadeiros positivos obtidos pelo algoritmo, pelo que automaticamente as métricas como o *recall* e e *f1 score* caíram para valores bastante pequenos. Relativamente à classificação de entidades para situações de *downgrade* os resultados obtidos foram os mesmos que aqueles que se têm vindo a observar ao longo dos diferentes testes realizados, sendo esta uma nova agregação e mesmo assim os resultados obtidos para este tipo de previsão foram similares aos demais, é expectável que os restantes algoritmos também eles tenham uma prestação dentro da norma.

Segue-se a análise ao algoritmo de logistic regression, cujos resultados constam na Figura 7.44, estes resultados indicam que o algoritmo não só teve uma prestação mais baixa que o algoritmo de averaged perceptron para a mesma agregação como também teve uma prestação inferior quando comprado com o mesmo algoritmo para o agregado correspondente que utiliza a coluna 'Date'. Assim sendo é claro que o uso deste algoritmo não se adequa à resolução do problema em questão.

No que toca ao algoritmo de bayes point machine este obteve resultados que indicam que o algoritmo não foi capaz de extrair informação útil a partir dos dados. A Figura 7.45 mostra traçado a azul o comportamento do classificador obtido com a união do algoritmo com a agregação de dados em questão. O baixo número de entidades classificadas com a classe positiva revela que o algoritmo tem alguma dificuldade em não atribuir o valor zero a entidades em avaliação, assim sendo é possível concluir que este é um algoritmo

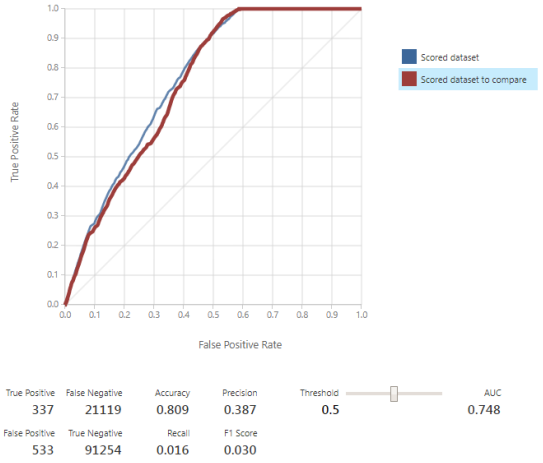


(a) Situação de Upgrade

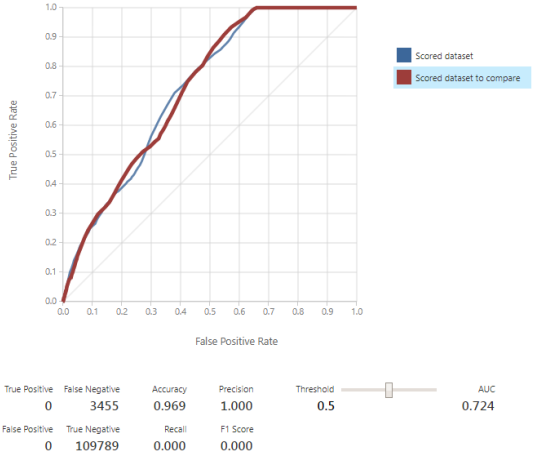


(b) Situação de Downgrade

Figura 7.43: Métricas de desempenho do algoritmo de Average Perceptron para *upgrade* e *downgrade* de pacotes, agregação por cliente e produto pela métrica ProcessedDateTime



(a) Situação de Upgrade



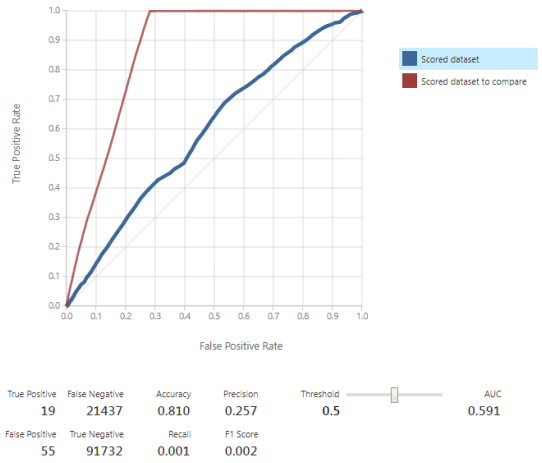
(b) Situação de Downgrade

Figura 7.44: Métricas de desempenho do algoritmo de Logistic Regression para *upgrade* e *downgrade* de pacotes, agregação por cliente e produto pela métrica ProcessedDateTime

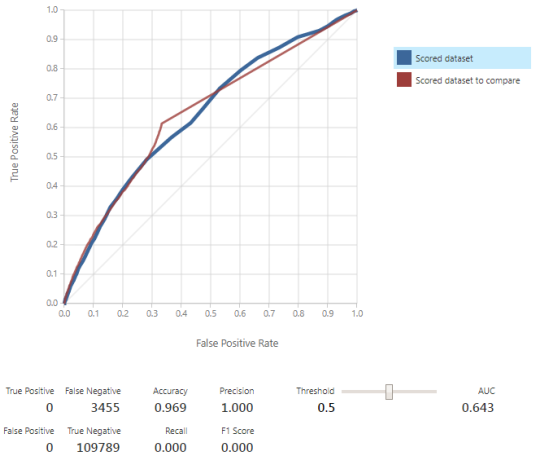
que não está apto a ser implementado na solução final deste projeto, uma vez que não conduz a resultados promissores.

Tal como havia acontecido com a agregação que tem em conta a informação presente na coluna 'Date' também aqui o algoritmo de decision forest teve a melhore prestação no que toca à classificação de entidades para situações de *upgrade*, do ponto de vista de coerência isto é bastante bom pois prova que com o mesmo estilo de conjunto de dados realmente os algoritmos tendem a comportar-se de formas diferentes, o que permite concluir que a aprendizagem não está a acontecer com recurso a elementos aleatórios nem produz resultados não determinísticos.

Os gráficos para os algoritmos de boosted decision tree (na Figura 7.47) e decision

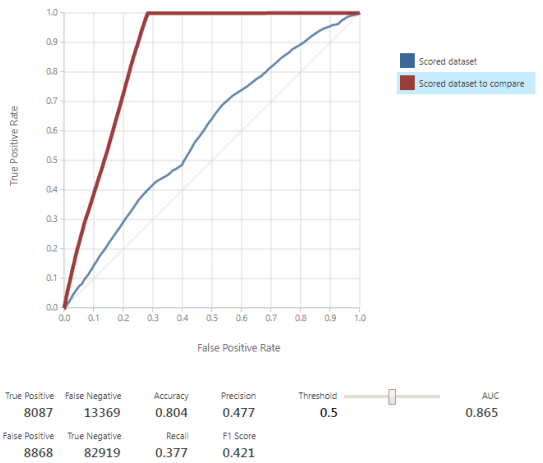


(a) Situação de Upgrade

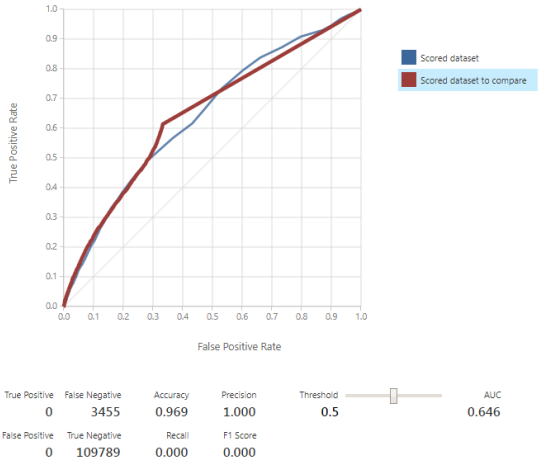


(b) Situação de Downgrade

Figura 7.45: Métricas de desempenho do algoritmo de Bayes Point Machine para *upgrade* e *downgrade* de pacotes, agregação por cliente e produto pela métrica ProcessedDateTime



(a) Situação de Upgrade

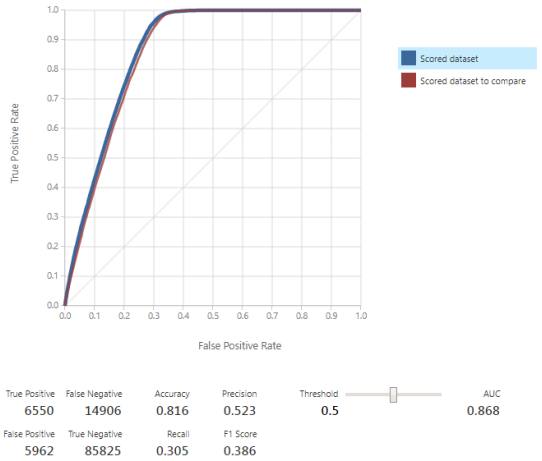


(b) Situação de Downgrade

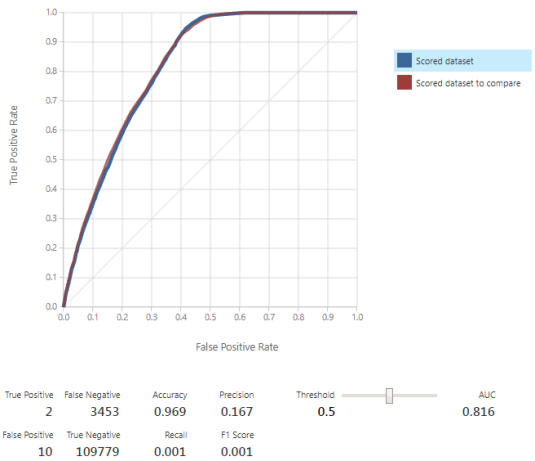
Figura 7.46: Métricas de desempenho do algoritmo de Decision Forest para *upgrade* e *downgrade* de pacotes, agregação por cliente e produto pela métrica ProcessedDateTime

jungle (presente na 7.48) mostram mais uma vez dois gráficos semelhantes, sendo que mais uma vez o primeiro apresenta uma prestação superior devido ao facto de classificar mais entidades de forma correta do que o segundo. Contudo surge uma surpresa no que toca à classificação de situações e *downgrade*, o algoritmo de boosted decision forest efetuou a classificação de entidades com o valor positivos, apesar deste apenas ter acertado duas das previsões positivas isto mostra que existiu algo nos dados que tornou possível ao algoritmo inferir a classificação com o valor 1. Com o aumento da quantidade de informação ao longo do tempo é interessante continuar a testar este algoritmo com este conjunto de dados no sentido de perceber se este vai adquirindo melhores resultados. Neste momento, para aplicação à solução do problema não é interessante do ponto de

vista de negócio pois as métricas de avaliação apresentam valores bastante diminuídos.

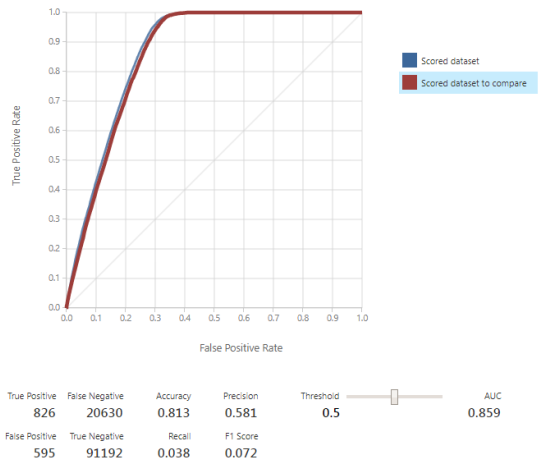


(a) Situação de Upgrade

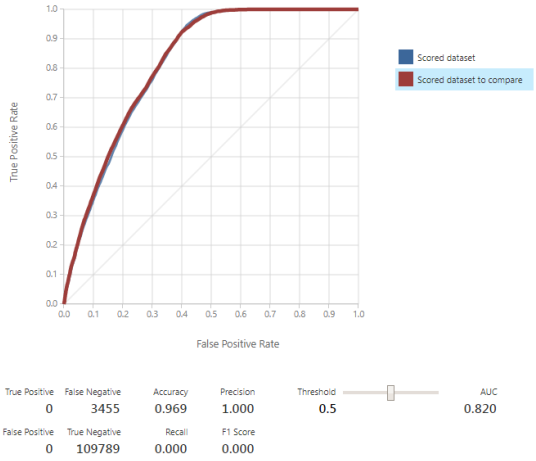


(b) Situação de Downgrade

Figura 7.47: Métricas de desempenho do algoritmo de Boosted Decision Tree para *upgrade* e *downgrade* de pacotes, agregação por cliente e produto pela métrica ProcessedDateTime



(a) Situação de Upgrade



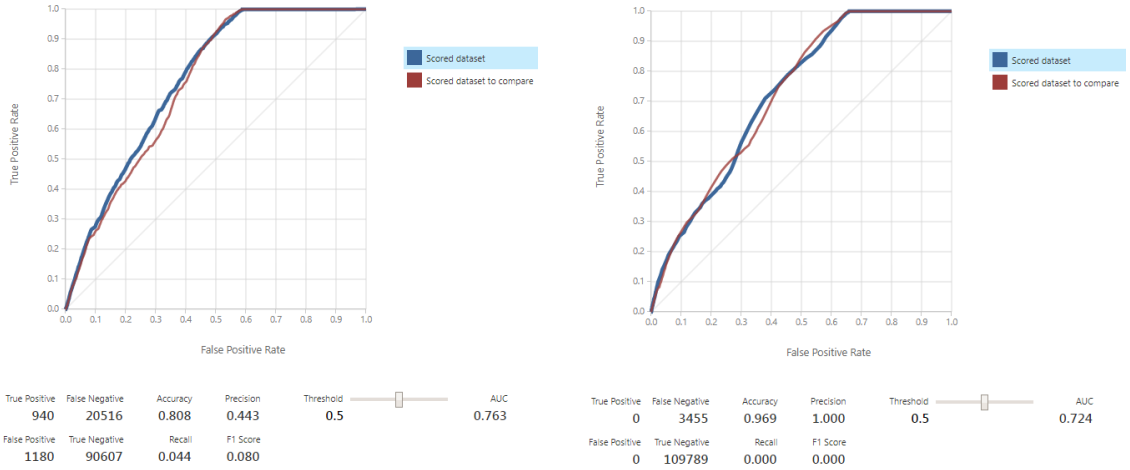
(b) Situação de Downgrade

Figura 7.48: Métricas de desempenho do algoritmo de Decision Jungle para *upgrade* e *downgrade* de pacotes, agregação por cliente e produto pela métrica ProcessedDateTime

7.1.2.6 Agregação pela métrica ProcessedDateTime, Cliente + Produto + Serviço

Mais uma vez como já era expectável os resultados da agregação com base na coluna ProcessedDateTime obtiveram resultados um pouco abaixo daqueles que foram os obtidos pela agregação que tem em conta a coluna Date, uma prova disto são os valores apresentados na Figura 7.49, uma vez que este apresenta resultados bastante inferiores àqueles que foram os obtidos pelo mesmo algoritmo com a agregação anteriormente analisada. Apesar de ainda faltarem testar mais algoritmos para esta agregação de dados a obtenção de zero entidades classificadas com a probabilidade de *downgrade* levou à necessidade de

perceber o que poderia influenciar todos os classificadores para todas as agregações a terem uma prestação tão pouco produtiva no que toca à deteção de situações de *downgrade*. Os frutos dessa procura e investigação encontram-se por isso explicados na Secção 7.3, onde são discutidos os desempenhos dos vários classificadores e justificações para tal.



(a) Situação de Upgrade

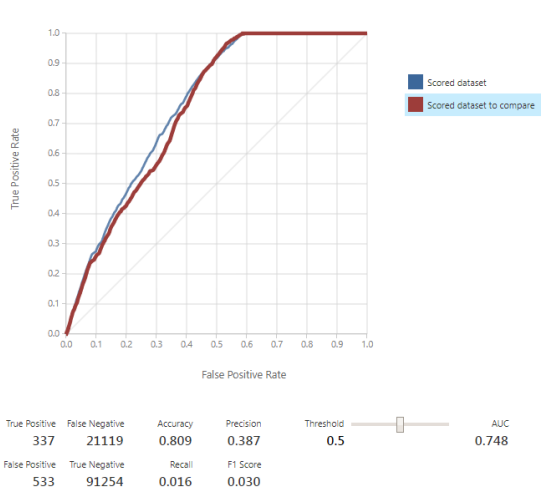
(b) Situação de Downgrade

Figura 7.49: Métricas de desempenho do algoritmo de Averaged Perceptron para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica ProcessedDateTime

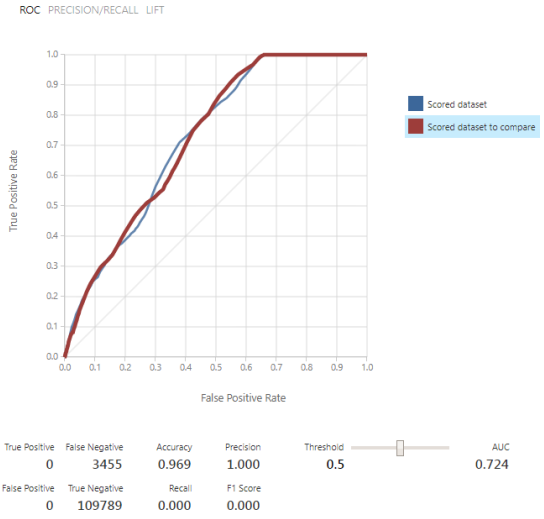
Os resultados provenientes da utilização do algoritmo de logistic regression encontram-se apresentados na Figura 7.50, em que apesar da curvatura desenhada ser idêntica à do averaged perceptron, bem como as métricas de *accuracy* e *precision*, é percetível que o algoritmo não tem um desempenho idêntico atendendo à queda de verdadeiros positivos quando comparado diretamente com o algoritmo de averaged perceptron, devido a este motivo as métricas de *recall* e *F1 score* viram o seu valor diminuído. Podemos também afirmar que atendendo ao facto de estas métricas apresentarem um valor tão baixo que estamos na presença de uma combinação de modelo que não é a mais favorável à utilização para a classificar de entidades no contexto do problema.

Para o algoritmo de bayes poins machine os resultados ficaram aquém das expetativas para as duas situações em análise, tanto previsão de *upgrades* como previsão de *downgrades*. Os resultados encontram-se explícitos na Figura 7.51 A Figura 7.51(b) mostra, sem surpresas, uma falta da habilidade por parte do classificador em identificar casos de *downgrades*, mas a surpresa relativa a este classificador refere-se ao gráfico presente na Figura 7.51(a). A baixa taxa de acertos relativos às entidades classificadas como verdadeiras positivas e a proximidade da linha (azul) do centro do gráfico mostra que a classificação por parte do modelo não é a melhor, os valores diminutos obtidos por este modelo descartam automaticamente qualquer hipótese de ponderação do seu uso no futuro para qualquer um dos problemas em mãos.

Como já vem a ser hábito nestas agregações de dados o algoritmo de decision forest

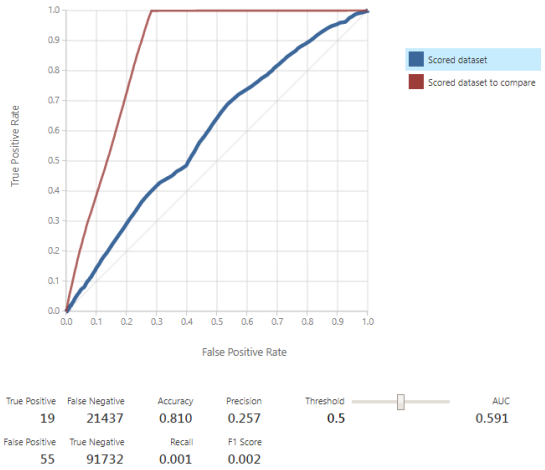


(a) Situação de Upgrade

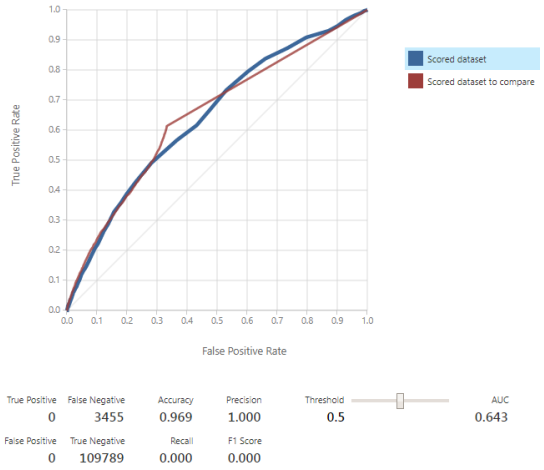


(b) Situação de Downgrade

Figura 7.50: Métricas de desempenho do algoritmo de Logistic Regression para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica ProcessedDateTime



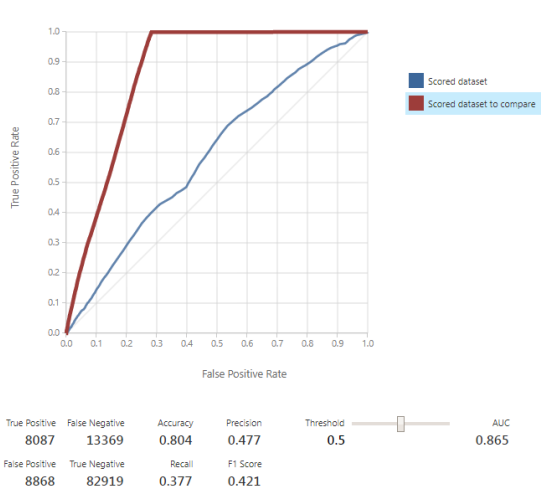
(a) Situação de Upgrade



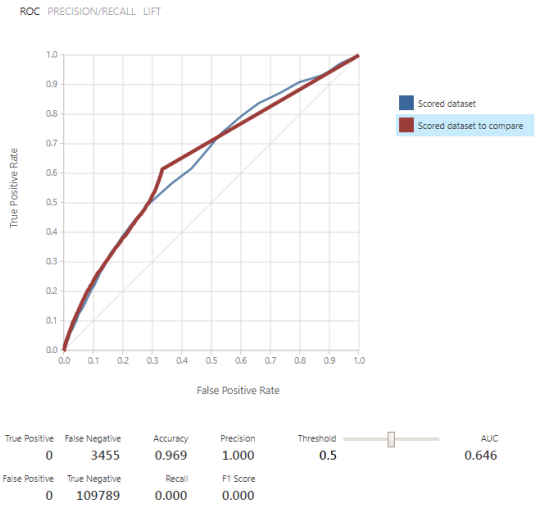
(b) Situação de Downgrade

Figura 7.51: Métricas de desempenho do algoritmo de Bayes Point Machine para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica ProcessedDateTime

demonstra (Figura 7.52) resultados bastante superiores aos apresentados por parte dos algoritmos estudados anteriormente para esta agregação especifica. Apesar de ainda assim existirem um total de falsos negativos superiores ao dobro daquele que é o número de verdadeiros positivos de previsões de situações de *upgrades* este é o algoritmo que garante melhores resultados para esta agregação de dados, pelo que é justo concluir que em caso de necessidade de utilização desta agregação em específico o algoritmo de decision forest seria a escolha clara a utilizar, devido ao facto de apresentar constantemente os melhores resultados nos testes efetuados.



(a) Situação de Upgrade



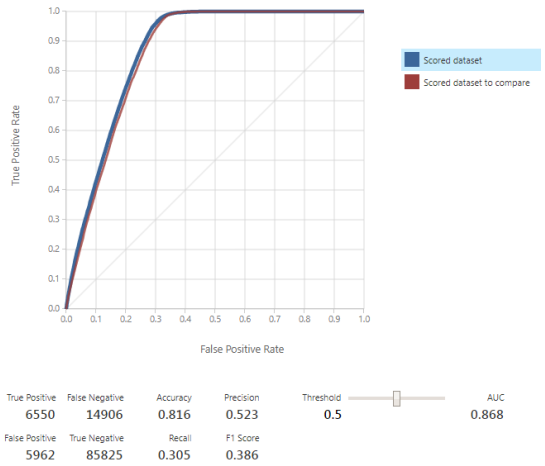
(b) Situação de Downgrade

Figura 7.52: Métricas de desempenho do algoritmo de Decision Forest para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica ProcessedDateTime

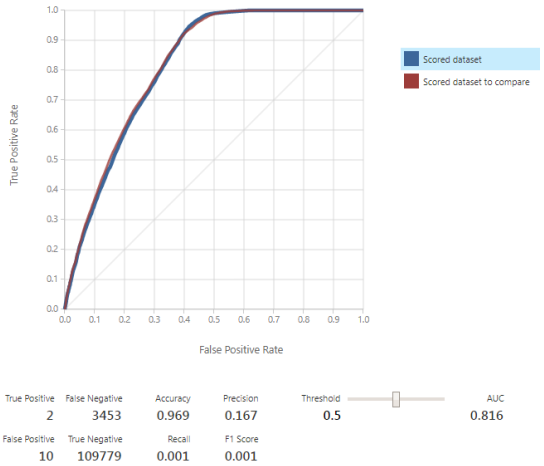
Para a agregação que tem em conta a coluna Date e utilizou a agregação de cliente, produto e serviço foi possível reparar que os algoritmos de boosted decision tree e de decision jungle obtiveram classificações bastante similares, contudo para esta agregação, que tem em conta a coluna de ProcessedDateTime para realizar a agregação isto não aconteceu, pelo contrário. Apesar de apresentarem visualmente curvas com desníveis semelhantes, como é visível através da avaliação das Figuras 7.53 e 7.54, os resultados observáveis através das outras métricas de avaliação mostram que estes dois modelos obtiveram desempenhos bastante diferentes. O facto da agregação baseada na coluna de ProcessedDateTime possuir poucos valores torna as pequenas variações nos dados muito mais influentes na classificação final, devido a este facto, algoritmos como o decision jungle tornam-se mais débeis, uma vez que os vários arcos para vários nós na árvore de decisão tendem a fazer variar o resultado de forma mais instável. Com a utilização de um conjunto de dados com mais entradas é expectável que os valores obtidos por via do algoritmo de decision jungle fossem mais parecidos com os obtidos com recurso ao algoritmo de boosted decision tree, como foi confirmado com a utilização da agregação por cliente, produto e serviço com base na coluna Date.

Ao completar esta avaliação fecha-se assim as possibilidades de modelos de aprendizagem automática para dois problemas. Passa a ser possível deliberar sobre qual o melhor algoritmo e agregação de dados para identificar potênciais situações *churn* relativos a dados de subscrições e passa também a ser possível averiguar qual o melhor modelo para dar resposta à necessidade de prever situações de *upgrades* e *downgrades* para estes dados. A averiguação de todas as possibilidades e decisão de qual a melhor opção a tomar no sentido de dar resposta ao problema será tomada na Secção 7.3.

A única pergunta que ainda não viu testada as diferentes hipóteses foi qual o melhor

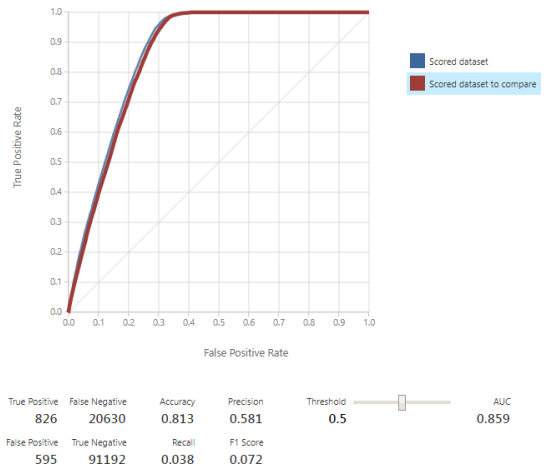


(a) Situação de Upgrade

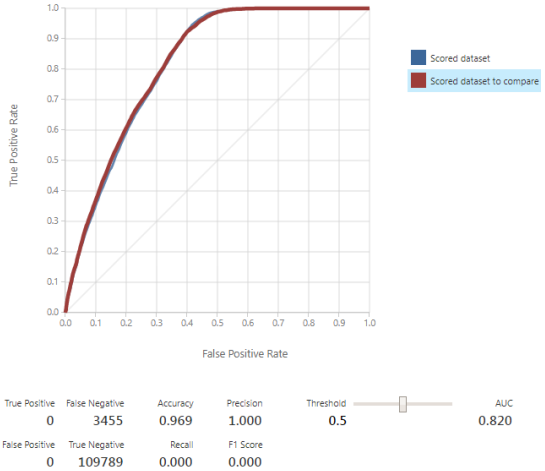


(b) Situação de Downgrade

Figura 7.53: Métricas de desempenho do algoritmo de Boosted Decision Tree para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica ProcessedDateTime



(a) Situação de Upgrade



(b) Situação de Downgrade

Figura 7.54: Métricas de desempenho do algoritmo de Decision Jungle para *upgrade* e *downgrade* de pacotes, agregação por cliente, produto e serviço pela métrica ProcessedDateTime

modelo para prever situações de *churn* quando se estão a utilizar dados faturados por utilização, sendo que essa pergunta será abordada de seguida, na Secção 7.2.

7.2 Testes para Dados de Utilização

No que toca aos dados de utilização, os dados utilizados para o treino dos modelos de aprendizagem foram os conjuntos de dados apresentados na Secção 5.2.2.1, sendo que estes foram testados com os diferentes algoritmos presentes na Secção 6.1. Devido à

natureza dos dados, não faz sentido propor diferentes pacotes a utilizadores de serviços faturados por utilização, uma vez que este tipo de serviço depende apenas da utilização que os clientes fazem dos serviços e não existem *upgrades* nem *downgrades* de pacotes associados a estes serviços, posto isto, o único tipo de modelo de aprendizagem automática criado a partir dos dados de utilização prende-se com a previsão de situações de *churn* por parte dos clientes.

7.2.1 Teste de algoritmos para classificação de churn

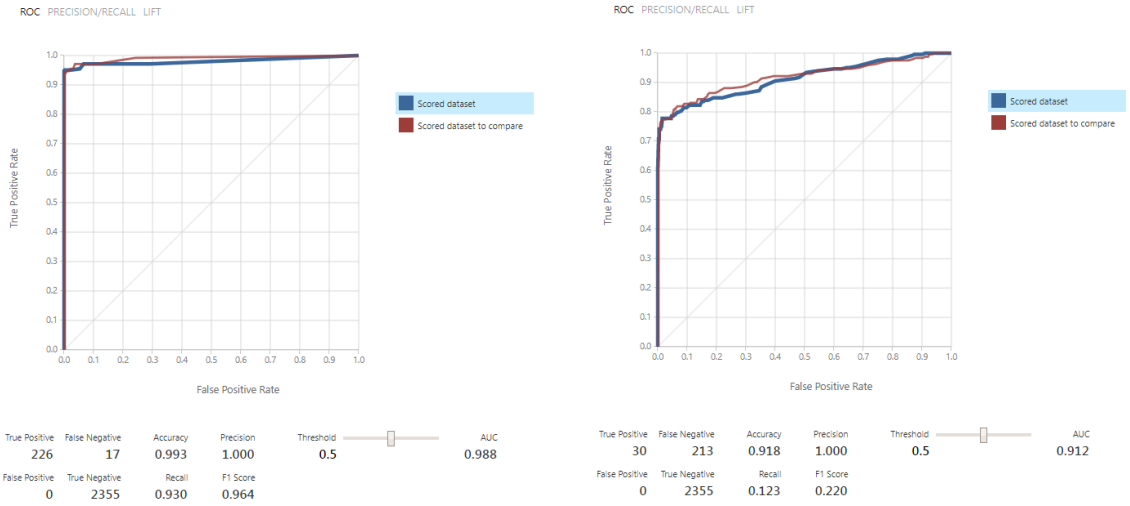
No que toca à análise realizada para os dados de utilização esta é realizada de forma independente para os dados que levam em conta os rácios de utilização e independentemente para os dados relativos às taxas, sendo que é objetivo deste estudo perceber que algoritmos proporciona o melhor resultado possível para cada uma das agregações e depois perceber que agregação permite detetar casos de *churn* com maior facilidade.

7.2.1.1 Agregação relativa aos rácios

Do ponto de vista da agregação que faz uso dos rácios de utilização dos serviços existe alguma diferença nos resultados obtidos para os algoritmos de averaged perceptron e logistic regression, como é possível confirmar na Figura 7.55. O desempenho do algoritmo de averaged perceptron proporcionou a identificação da grande maioria dos casos em que houve uma situação de *churn*, ao contrário do algoritmo de logistic regression onde apenas uma pequena parte destes foi identificada, com este resultado podemos concluir que foi fácil de alguma forma traçar uma linha na qual se dividiu as entidades em risco de *churn* das demais, enquanto que para o algoritmos de logistic regression não foi possível determinar a melhor probabilidade para cada métrica de acordo com o objetivo. Para estes dois algoritmos resta apenas confirmar se o desempenho dos mesmos manteve esta vantagem comparativa quando utilizado o conjunto de dados relativo ao uso de taxas do serviço.

Passando para a análise dos algoritmos de Bayes Point Machine e Decision Forest representados na Figura 7.56, a utilização do algoritmo representado na Figura 7.56(a) possibilitou a observação de um algoritmo ligeiramente melhor do que o de averaged perceptron, apesar das diferenças nas avaliações de métricas serem bastante baixas, esta diferença mantém-se percentualmente diferente ao aumentar o conjunto de dados, ou seja, ao longo do tempo à medida que forem sendo extraídos mais e mais dados sobre os clientes é de esperar que com a utilização do algoritmo de bayes point machine seja possível obter uma maior melhoria dos resultados do se fosse utilizado o algoritmo de averaged perceptron. Para o algoritmo de decision forest este teve claramente uma prestação inferior ao algoritmo anterior, pelo que fica automaticamente excluído das hipóteses a ponderar para implementação.

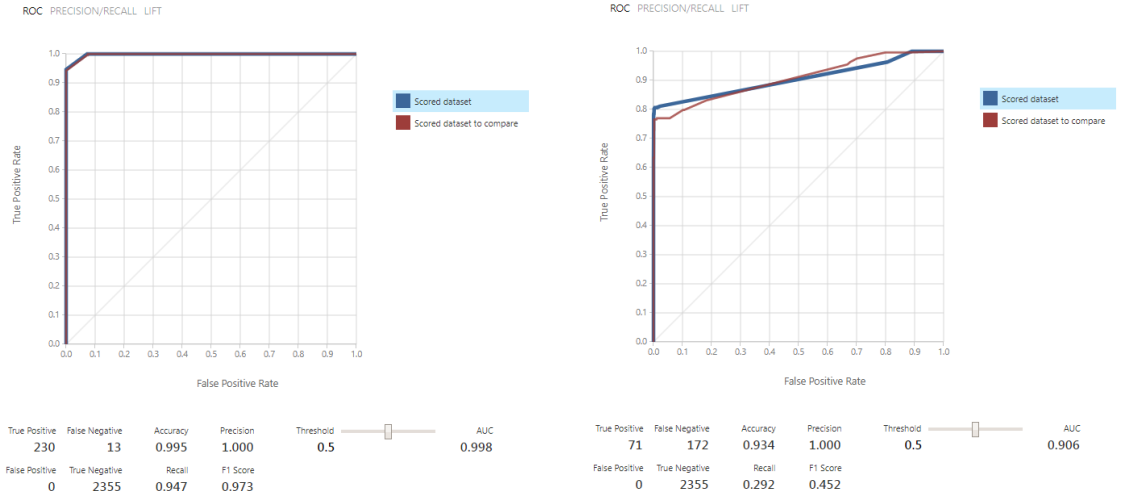
Dos gráficos presentes na Figura 7.57 deve-se considerar apenas as linhas a azul, tanto



(a) Averaged Perceptron

(b) Logistic Regression

Figura 7.55: Métricas de desempenho do algoritmo de Averaged Perceptron e Logistic Regression para dados de rácio



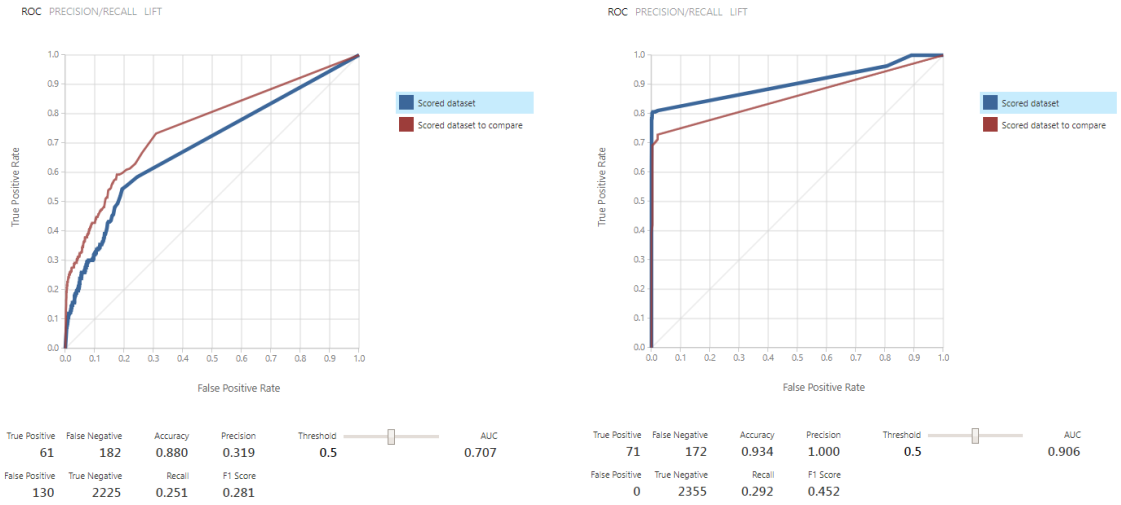
(a) Bayes Point Machine

(b) Decision Forest

Figura 7.56: Métricas de desempenho do algoritmo de Bayes Point Machine e Decision Forest para dados de rácio

para o algoritmo de boosted decision tree presente na Figura 7.57 (a) como para o algoritmo de decision jungle presente na Figura 7.57 (b). Estes dois algoritmos, juntamente com o algoritmo de decision forest são algoritmo que derivam da utilização de árvores de decisão, sendo que o resultado do algoritmo de decision jungle é exatamente equivalente ao de decision forest. Estes três algoritmos não obtiveram as melhores previsões pois como estamos a utilizar um conjunto de dados com taxas de utilização de serviços acaba por ser difícil de implementar recorrendo a árvores de decisão, pois uma pequena mudança percentual pode influenciar muito no resultado final. Assim sendo estes três algoritmos acabam por ficar aquém das expetativas no que toca à previsão de situações

de *churn*, uma vez que existem candidatos melhores.



(a) Boosted Decision Tree (b) Decision Jungle

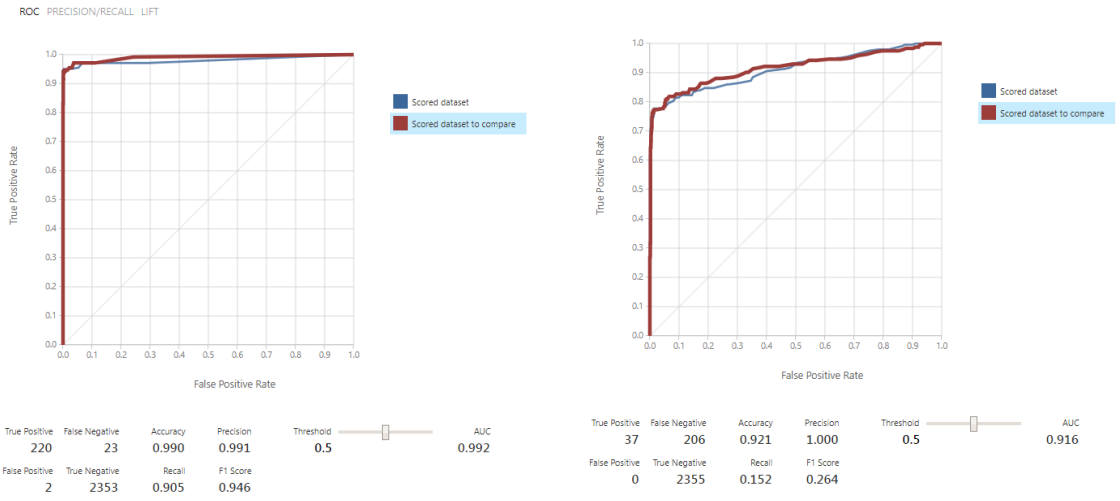
Figura 7.57: Métricas de desempenho do algoritmo de Boosted Decision Tree e Decision Jungle para dados de rácio

7.2.1.2 Agregação relativa às taxas

As duas agregações realizadas com os dados de utilização revelaram-se bastante similares do ponto de vista de rendimento por parte dos diferentes algoritmos, de forma geral os mesmos algoritmos obtiveram o mesmo desempenho para as duas agregações de dados, apresentando apenas algumas diferenças.

Como é possível verificar na Figura 7.58 os resultados obtidos pelos algoritmos average perceptron e logistic regression são bastante idênticos aos obtidos por parte dos mesmos algoritmos mas para a agregação de dados referente aos rácios de utilização, contudo é de notar que o número de falsos negativos e falsos positivos subiram, quando comparamos estes dados com os outros, fazendo assim com que métricas de avaliação desçam ligeiramente.

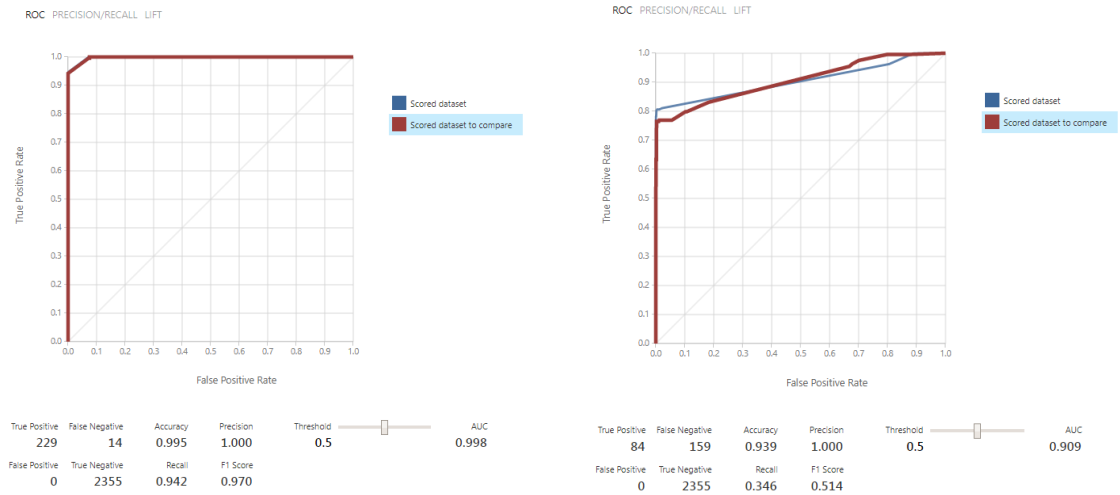
Mais uma vez o algoritmo de bayes point machine obteve a melhores resultados para o problema da deteção de *churns* para os dados de utilização, embora a diferença seja pequena, este continua a ter uma prestação ligeiramente superior à do algoritmo de average perceptron. Podemos verificar na Figura 7.59, o número de casos de possíveis situações de *churns* não identificado é baixo, o que é bastante positivo do ponto de vista do negócio, pois existe uma pequena percentagem de clientes sobre a qual não foram ativados os meios que têm como objetivo captar o cliente e evitar que este deixe de ser cliente. Com esta informação relativa às duas agregações podemos afirmar que para os dados faturados por utilização o algoritmo de bayes point machine apresenta resultados fortemente positivos. Por outro lado o algoritmo de decision forest não apresenta valores tão bons uma vez que apresenta um valor de falsos negativos superior ao de verdadeiros positivos.



(a) Average Perceptron

(b) Logistic Regression

Figura 7.58: Métricas de desempenho do algoritmo de Average Perceptron e Logistic Regression para dados de taxas

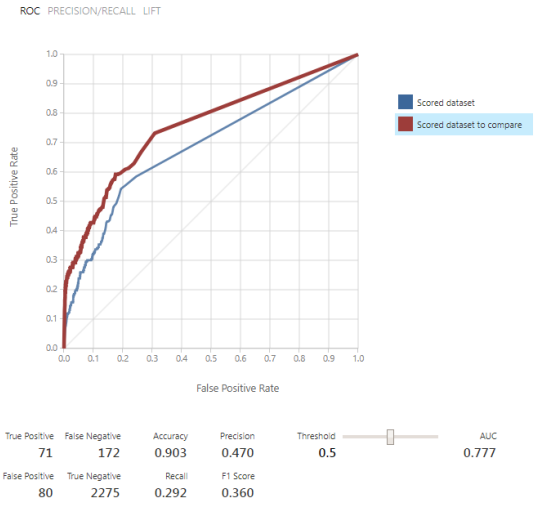


(a) Bayes Point Machine

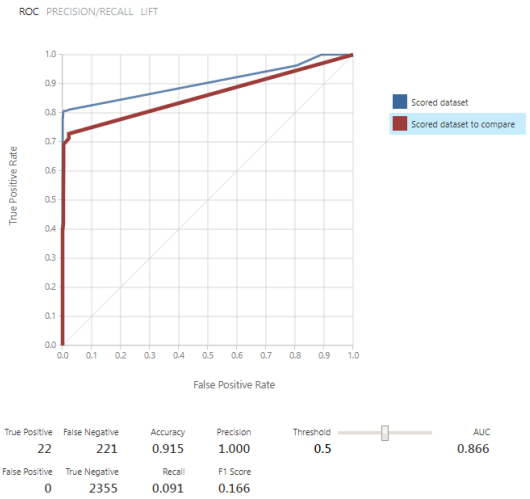
(b) Decision Forest

Figura 7.59: Métricas de desempenho do algoritmo de Bayes Point Machine e Decision Forest para dados de taxas

De forma análoga ao algoritmo de decision forest também o algoritmo de boosted decision tree e decision jungle, apresentado na Figura 7.60, apresentam valores que não são os mais desejados para um classificador, provando assim que algoritmos que se baseiam em árvores de decisão não são os melhores para classificar este tipo de dados. O facto da análise às diferentes possibilidades entre agregações e algoritmos ter sido metódica e exaustiva permitiu perceber e comprovar que os mesmos algoritmos perante conjuntos de dados semelhantes acabam por ter os mesmos resultados, nunca fugindo muito do ponto de vista de métricas de avaliação como é o caso da *accuracy e precision*.



(a) Boosted Decision Tree



(b) Decision Jungle

Figura 7.60: Métricas de desempenho do algoritmo de Boosted Decision Tree e Decision Jungle para dados de taxas

Uma vez realizados todos os testes para os diferentes algoritmos e diferentes agregações de dados criadas, passa a ser possível deliberar sobre qual a melhor combinação para implementar a solução que dará resposta ao problema, este tipo de análise e discussão será abordada na Secção 7.3.

7.3 Escolha dos Algoritmos e Conjuntos de Dados

De forma a que seja possível comparar de forma mais eficiente as diferenças entre combinações das diferentes agregações de dados e algoritmos realizadas, encontram-se em Apêndice as Tabelas de comparação relativas aos dados de subscrições para deteção de *churns* (Apêndice C), deteção de situações de *upgrade* (Apêndice D), detenção de situações de *downgrade* (Apêndice E) e deteção de situações de *churn* para dados de utilização (Apêndice F).

Começando pela análise do problema da previsão de *churn* para os dados de subscrições podemos facilmente perceber que existiram conjuntos de dados que obtiveram uma prestação bastante inferir aos demais. As duas agregações elaboradas que levavam em conta a coluna de ProcessedDateTime apresentaram valores de avaliações nas métricas bastante inferiores às outras agregações. Apesar do algoritmo de BoostedDecisionTree ter suscitado bons resultados no panorama geral de um classificador, este continuou aquém do esperado, sendo que existem melhores resultados obtidos por este mesmo algoritmo para outras agregações. A baixa prestação obtida pelas agregações que fazem uso da coluna de ProcessedDateTime fica a dever-se maioritariamente à quantidade inferior de dados que estas dispunham, o facto de existirem menos valores únicos na coluna ProcessedDateTime, devido ao espaçamento temporal da extração de dados, leva a que existam menos dados disponíveis e que seja mais difícil para os algoritmos de aprendizagem

automática detetarem padrões e relações entre os dados, culminando assim em classificadores menos aptos. Posto isto podemos retirar da hipótese de uso as duas agregações em questão, diminuindo assim o conjunto de possibilidades.

Relativamente às outras agregações feitas, é possível observar que a agregação que leva em conta os clientes e suas subscrições ao longo do tempo obteve um desempenho inferior às restantes. O facto das outras agregações terem produzido resultados perfeitos no que toca a algoritmos como o averaged perceptron, logistic regression, bayes point machine e boosted decision tree leva a que esta agregação fique abaixo das demais. Com a exclusão da hipótese de utilizar a agregação de cliente e subscrições para a implementação final do projeto restam assim três possibilidades de agregações. Os resultados obtidos ao longo dos testes revelaram que tanto a agregação apenas por cliente como as duas agregações que levam em consideração a coluna Date obtiveram resultados excelentes do ponto de vista dos classificadores, chegando a empatar no seu desempenho. Assim sendo, qualquer uma das três possibilidades será viável para implementação final, contudo, como é preferível que se utilize apenas um modelo único, no sentido de facilitar o uso e compreensão de como a solução funciona, os resultados para as questões de previsão de *upgrades* e *downgrades* irão influenciar também sobre que modelo utilizar para a solução dos *churns*.

A análise dos modelos para o problema dos casos de *upgrade* de serviços encontra-se disponibilizados no Apêndice D. É possível reparar que para este problema existiu uma quantidade menor de modelos aptos a resolver a questão, esta diferença deve-se naturalmente por se tratar de um problema diferente e existirem menos casos de *upgrades* de serviços do que de situações de *churn*, sendo assim este não é comparável diretamente com o problema anterior. No contexto deste problema houve claramente três agregações que se destacam das demais, sendo elas a agregação por cliente, a agregação por cliente e subscrição e a agregação por subscrição, produto e serviço baseada na coluna Date. Destas três agregações duas são comuns às agregações que obtiveram melhores resultados para o problema de *churns*, assim sendo, com vista a minimizar a complexidade da solução a agregação por cliente e subscrição acaba por ser excluída. Relativamente às duas agregações restantes estas não obtiveram resultados idênticos. A agregação por cliente apenas demonstrou não só obter bons resultados com mais classificadores como também apresentou resultados melhores para algoritmos específicos quando comparada diretamente com a agregação de subscrição, produto e serviço.

A boa prestação do modelo composto pelo algoritmo de bayes point machine e pela agregação de cliente apenas torna-o elegível a ser utilizada na solução final do problema, uma vez que foi este modelo que proporcionou a melhor solução de entre todos os modelos testados. Assim sendo, e como este modelo foi também um dos modelos que obteve uma classificação mais acertada para o problema dos *churns*, passa a ser possível concluir sobre que agregação de dados e que algoritmo utilizar para dar resposta aos problemas. Tanto para o problema de deteção de *churns* como para o problema de deteção de casos de *upgrades* de pacotes, uma agregação mais macroscópica revelou-se vantajosa para dar solução aos problemas.

Relativamente ao problema da previsão de situações de *downgrades* de pacotes o panorama de deteção não foi o melhor, como é possível confirmar na informação presente no Apêndice E. Os modelos de aprendizagem automática não foram capazes de relacionar os dados em função da classe a classificar para as entidades dos conjuntos de dados, isto ficou a dever-se ao facto de que quando comparamos a quantidade de existências de *upgrades* com os *downgrades* o número de *downgrades* existentes no sistema é bastante diminuto. A baixa quantidade de *downgrades* torna-se um entrave à aprendizagem pois os modelos necessitam de uma base de conhecimento representativa de forma a conseguir relacionar as instâncias, é expectável que com o decorrer do tempo e a extração continuada de mais dados que possuam a existência de *downgrades* os algoritmos venham a apresentar melhores resultados. Contudo do ponto de vista do negócio também faz sentido a baixa existência de *downgrades*, fazendo também a comparação com o negócio das telecomunicações, uma vez que a dinâmica de negócio é semelhante, normalmente não é sugerido a um cliente que efetue um *downgrade* ao seu serviço caso este não esteja a utilizar a totalidade do mesmo, o caso mais comum passa por deixar o cliente utilizar o serviço que tem subscrito sem que haja interferência no processo. Este tipo de abordagem culmina mais tarde na existência de um *churn*, potenciando assim o mecanismo de previsão de *churns* mas prejudicando a previsão de *downgrades*. Caso seja objetivo modificar estar abordagem ao negócio é necessario que se continue a extrair dados ao longo do tempo, mesmo que exista uma percentagem baixa de existencia de *downgrades* até ao ponto em que os algoritmos já conseguem relacionar os dados e garantir alguma fiabilidade no que toca à previsão de classes. Depois deste ponto passa a ser possível sugerir a clientes que efetuem *downgrades*, o que vai fazer aumentar o número de *downgrades* existentes no sistema e beneficiar mais uma vez os mecanismos de deteção dos mesmos.

Do ponto de vista de implementação de uma solução que garanta resultados fiáveis, para os problemas de previsão de *churns* e *upgrades*, conclui-se que o melhor modelo a utilizar passa por combinar a agregação dos dados dos clientes apenas com o algoritmo de bayes point machine. A utilização de um único modelo para ambos os problemas proporciona uma implementação simplificada, bem como permite uma documentação da solução mais simples e direta. As vantagens associadas a esta opção garantem que a passagem de contexto de como a solução funciona e do que necessita seja mais direta, simples e rápida, evitando assim que a implementação se torne demasiado complexa para fontes externas.

Relativamente à previsão de casos de *downgrade* foi também possível concluir duas coisas, a primeira é que os algoritmos não conseguem, neste momento, detetar estas ocorrências, contudo do ponto de vista do negócio atual é normal e faz todo o sentido que isto aconteça, pelo que este tipo de resultado prova que a solução encontrada está de facto consistente com o problema de negócio no qual se insere.

Para os dados de utilização as duas agregações em estudo obtiveram os mesmos resultados, como é possível verificar através da análise da Tabela disponibilizada no Apêndice F. As conclusões retiradas dos testes revelaram que o algoritmo de bayes point machine

conseguiu obter melhores resultados para as agregações de dados relativas a dados de utilização. Um dos pontos importantes sobre a boa prestação deste algoritmo é o facto deste ter em atenção o desnível de classes que existe, o facto de existir muito menos situações de churn nos dados beneficia a prestação do bayes point machine por este ter em base um modelo probabilístico no seu funcionamento, adquirindo assim melhores resultados que os demais. Do ponto de vista de implementação é viável utilizar qualquer uma das duas agregações de dados criadas para o algoritmo de bayes point machine. Será utilizada a agregação relativa às taxas do serviço pelo facto desta possuir uma maior quantidade de dados de faturação, como foi indicado pela equipa dedicada ao Cloud Cockpit, contudo é útil continuar a avaliar estas duas possibilidades no futuro, pois à medida que vão sendo extraídos mais dados o desempenho entre as duas agregações pode mudar, passando assim a ser mais favorável utilizar a outra agregações de dados.

7.4 Sumário

Neste capitulo foi descrito em detalhe todas as etapas necessárias à escolha de como implementar a solução do problema. Numa primeira parte foi necessário descrever todos os algoritmos que estão por detrás dos modelos de aprendizagem automática para a classificação de entidades que serão utilizados no projeto. De forma a perceber que algoritmo garante os melhores resultados foi necessário observar o comportamento e variação de algumas métricas de avaliação, métricas estas também identificadas e explicadas nesta secção.

Por fim, uma vez explicados todos os intervenientes que participam no treino e seleção de modelos foi descrita a arquitetura do ambiente de testes, arquitetura esta que possibilitou a comparação exaustiva de todas as possibilidades de modelos de aprendizagem automática. Após recolher os dados de todos os testes passou a ser possível concluir sobre os mesmos e escolher qual o modelo a ser implementado na solução final do projeto.

No próximo capítulo utilizar-se-à o modelo escolhido de forma a finalizar um fluxo de ações que possibilite a classificação de entidades. Após este fluxo estar criado passa a ser possível disponibilizar um *web service* para que o serviço seja utilizado por entidades exteriores, sendo que a informação relativa a esta parte apresenta-se de seguida no Capítulo 8.

Capítulo 8

Implementação de um Web Service

Após ter sido decidido qual o melhor algoritmo a aplicar sobre o melhor conjunto de dados para cada objetivo do projeto, o próximo passo é implementar a possibilidade de utilizar os modelos de aprendizagem automática treinados para a classificação de novas entidades. Isto porque no contexto do Azure Machine Learning Studio é possível utilizar o fluxo de módulos criado para fazer a classificação de entidades diretamente no Studio, contudo torna o processo algo demorado e dependente de um série de etapas no que toca ao processo de importação de novos dados para o fluxo, etapas essas que são realizadas manualmente e portanto levam algum tempo a serem concluídas.

Com o objetivo de otimizar o processo de classificação de novas entidades, foi disponibilizado um *web service* de forma a que o mesmo possa ser utilizado por aplicações externas de forma a automatizar o sistema. Neste capítulo encontra-se primeiramente uma descrição de como foi efetuada a disponibilização do *web service* e quais as suas potencialidades. Numa segunda parte será descrito o trabalho desenvolvido em torno da criação de uma aplicação que efetua pedidos *batch* ao *web service* previamente montado e que permite assim a ligação entre diferentes ferramentas Microsoft de forma a satisfazer todas as necessidades do projeto, permitindo uma solução integrada.

8.1 Disponibilização de um Web Service

Assim que o fluxo de módulos no Azure Machine Learning Studio estiver totalmente montado passa a ser possível disponibilizar um *web service* que permite a entidades externas aceder e utilizar o fluxo, efetuando pedidos ao mesmo. De forma a que a disponibilização desta funcionalidade se processe de forma simples, a própria ferramenta do AML Studio permite ativar a funcionalidade de *web service*. Ao ativar esta funcionalidade é criada uma réplica do fluxo original, onde é necessário efetuar algumas modificações aos módulos

com o objetivo de classificar novas entidades.

Todos os módulos que são responsáveis pela modelação dos dados inicias não poderão sofrer qualquer modificação, isto prende-se com o facto dos modelos de aprendizagem automática terem sido treinados para uma estrutura de dados especifica, assim sendo, se esta estrutura sofrer qualquer alteração a nível do número de colunas, a tipologia dos dados presentes nas colunas, entre outras, será levantada uma exceção no fluxo. Contudo existem alguns módulos que têm de ser retirados, no fluxo original após se efetuar a modelação de dados necessária o fluxo dava entrada no módulo *split data* e seguidamente passavam para o módulo de *train model*. Estes dois módulos deixam de fazer sentido do ponto de vista de um *web service*, isto deve-se ao facto do modelo de classificação já estar treinado e pronto para classificar entidades, assim sendo, quando a informação estiver totalmente modelada esta passa diretamente para o módulo de *score model*, cujos *inputs* são justamente o conjunto de dados a classificar e o classificador previamente treinado. Após este momento os dados estão classificados de acordo com o modelo escolhido e não precisam de mais nenhum tipo de tratamento ou operação, sendo que podem ser conectados ao módulo de *web service output* de forma a que estes sejam enviados para o processo que evocou a operação de classificação.

Uma vez criada a possibilidade do fluxo ser acedido por uma fonte externa, o próprio Azure Machine Learning Studio cria uma página na diretoria da conta associada ao fluxo do *web service* com informação relativa ao *web service*. Esta página contem informação que é necessária ao acesso do processo de classificação por terceiros. Para alem da chave da API são também criados dois *endpoints* para realizar pedidos ao *web service*, um para pedidos do tipo pedido/resposta, em que é enviada uma única entidade e o modelo retorna a classificação para essa entidade singular e um segundo para execuções *batch*, ou seja, para pedidos que contenham grandes quantidades de dados a classificar. Com estas informações disponíveis passa a ser possível criar uma aplicação que permita utilizar o serviço criado no Azure Machine Learning Studio, sendo que este tópico é abordado na Secção 8.2.

8.2 Criação de uma Aplicação Consumidora

A criação de uma aplicação que faz uso da API disponibilizada pelo *web service* do fluxo montado no Azure Machine Learning Studio nasceu da necessidade de automatizar o processo de classificação de entidades. Caso esta não fosse criada seria necessário abrir o Studio, definir a proveniência dos dados de forma a que estes fossem importados para o fluxo, iniciar o processo, esperar que o mesmo terminasse e depois definir para onde enviar os dados, tudo manualmente. De forma a que o processo seja mais rápido e com menos etapas surgiu a criação de uma aplicação em C# que interage com diferentes ferramentas da Microsoft.

Inicialmente os dados do problema eram acedidos através da utilização da ferramenta SQL Server Management Studio, infelizmente não foi possível criar uma ponte direta em

que a aplicação evocasse a informação do SSMS diretamente para o *web service* do Azure Machine Learning Studio e depois canalizasse o *output* para outro destino, pois o *web service* não o permite. Assim sendo, por forma a resolver o problema, foi decidido que se faria uso de um *blob storage* em Azure para armazenar a informação tanto de *input* como de *output*. Um *blob storage* está otimizado para armazenar grandes quantidades de dados, sendo que este é também bastante útil no armazenando de ficheiros para acesso distribuído.[12]

O funcionamento da aplicação criada pode ser distinguido em três etapas. A primeira passa por importar os dados iniciais para um blob, sendo que para isto o utilizador apenas necessita de fornecer as credenciais da conta Azure onde o blob se localiza bem como as credenciais da base de dados a importar e a aplicação trata do resto da importação. Uma vez carregado todo o ficheiro com o conjunto de dados inicial a aplicação evoca a API do *web service* e envia a informação presente no blob para que esta seja classificada. Após o modelo de aprendizagem automática acabar de classificar as entidades enviadas, a aplicação é responsável por direcionar estes resultados para um novo ficheiro, também este armazenado em blob azure.

Neste momento a execução da aplicação termina, uma vez que os dados classificados já deram entrada no blob de destino. A partir deste momento é possível utilizar os dados finais não só para o efeito a que eles se destinam, de identificar situações de intervenção, quer este seja de casos e *churn* quer seja de casos de sugestão de *upgrades* e *downgrades* mas também é possível utilizar os dados em qualquer ferramenta de visualização ou modelação de dados, de forma a potenciar o valor dos dados.

Com a criação desta aplicação todo o processo desde o armazenamento, recolha, classificação e utilização dos dados fica disponível na cloud, potenciando assim a disponibilidade dos dados e dos serviços. Uma vantagem dos dados finais ficarem alojados em blobs azure, para alem da vantagem clara de permitirem um acesso distribuído, é também a flexibilidade de importação para outras ferramentas, o facto das ferramentas Azure permitirem uma integração facilitada não só com todas as ferramentas Microsoft mas também com diferentes ferramentas de outros fornecedores facilitará a utilização da informação a partir deste ponto. Assim sendo estamos na presença de um sistema funcional, de uso facilitado e que nunca depende das características da máquina do utilizador.

8.3 Sumário

Neste capítulo apresenta-se a implementação de um *web service* e das suas potencialidades para o desenvolvimento da solução. É abordado o tópico relativo à criação de uma aplicação que utiliza o *web service* disponibilizado de forma a realizar pedidos para classificação de entidades em grandes quantidades de forma automática. Esta aplicação faz a interligação de diferentes ferramentas Azure, de forma a permitir que o armazenamento da informação apenas dependa da cloud e não de máquinas em particular. Por fim foi

também abordado o facto desta informação já classificada poder ser integrada com ferramentas de visualização de forma simples e rápida, aumentando assim o valor associado a esta.

No próximo capítulo dá-se a conclusão desta tese, onde serão abordadas diferentes conclusões que foram sendo retiradas ao longo do desenvolvimento do projeto e apresentados alguns pontos relativos a possíveis trabalhos futuros.

Capítulo 9

Conclusão e Trabalho Futuro

9.1 Conclusão

A aplicação de técnicas de aprendizagem automática tem emergido como uma estratégia para endereçar limitações existentes no mercado. À medida que o número de equipamentos cresce e com eles cresce também a quantidade de informação a ser gerada, surge a necessidade de processar esta informação de forma útil e escalável. A informação ganha mais valor se esta for utilizada com diferentes objetivos para além do seu armazenamento e monitorização, pelo que a previsão de resultados foi também uma aplicação explorada neste trabalho.

Existem diferentes formas e plataformas disponíveis para endereçar a previsão de resultados. Neste trabalho em concreto foram utilizadas tecnologias Microsoft, das quais se deve realçar o uso da plataforma de Azure Machine Learning Studio, uma vez que esta demonstrou ser uma plataforma poderosa no que toca à importação, tratamento e modelação de informação bem como na aplicação de técnicas de aprendizagem automática para classificação de entidades.

Para alcançar o objetivo de prever situações de desistência por parte de clientes, bem como para prever situações de mudanças de pacotes estudaram-se diferentes formas de abordar o problema, passando pela aprendizagem supervisionada, os algoritmos de classificação entre duas classes, as métricas de avaliação de classificadores e os próprios módulos internos disponibilizados pela plataforma de Azure Machine Learning Studio.

Foi possível criar e testar diferentes tipos de modelos de aprendizagem automática, concluindo assim que tipo de modelo mais se adequava à solução do problema em questão. Os resultados obtidos por parte dos classificadores mostraram-se bastante bons, estando estes adequados ao problema e à informação em causa. De forma a criar uma solução que possibilitasse a utilização num ambiente automatizado, foi desenvolvido um fluxo

de operações que culminaram na disponibilização de uma API de forma a que este fluxo possa ser integrado em soluções maiores, tirando assim o máximo partido do mesmo.

9.2 Trabalho Futuro

Nesta Secção apresentamos algumas sugestões de trabalho futuro, consequentes do projeto desenvolvido ao longo deste ano.

Extração de dados: Relativamente à extração de dados é importante que esta continue a ocorrer periodicamente. O aumento da informação disponível melhora a prestação dos classificadores, pois com mais informações relativa aos dados existe uma maior cobertura de eventuais situações que ocorrem ao longo do tempo. Assim sendo, é de extrema importância que se continue a obter dados de forma a que os dados em uso não se tornem ultrapassados e que seja possível obter resultados melhores ao longo do tempo.

Treino de modelos: De forma a acompanhar a evolução da utilização de serviços é necessário que os modelos de aprendizagem automática sejam treinados periodicamente. Isto deve-se ao facto de que os dados que hoje são úteis e permitem concluir sobre o negócio podem vir a mudar. A alteração de consumo dos clientes em alguns produtos pode sofrer variações ao longo do tempo, para evitar que os classificadores se tornem antiquados é necessário voltar a treinar os modelos de forma a que estes utilizem os dados mais recentes para tirar conclusões sobre os mesmos e continuem atuais.

Visualização de resultados: Com a informação processada e classificada a partir do fluxo criado nesta solução é possível gerar visualizações de dados para tornar mais elucidativo o que influencia a classificação. É possível criar visualizações que mostrem quais os produtos em que existe uma maior taxa de desistência, que tipos de custos estão associados a estes serviços, entre outras. Uma vez que toda a solução está disponível com base na *cloud* e permite uma integração fácil com diferentes serviços, um trabalho futuro a aplicar por cima do trabalho realizado até à data passa pela criação de visualizações interessantes utilizando a informação classificada.

Bibliografia

[1] AltexSoft. *Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watson*. Last accessed 5 July 2019. 2019. URL: https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/.

[2] S. S. Anand e B. Mobasher. "Intelligent techniques for web personalization". Em: *Proceedings of the 2003 international conference on Intelligent Techniques for Web Personalization*. Springer-Verlag. 2003, pp. 1–36.

[3] C. Basu, H. Hirsh, W. Cohen et al. "Recommendation as classification: Using social and content-based information in recommendation". Em: *Aaai/iaai*. 1998, pp. 714–720.

[4] A. Bhande. *What is underfitting and overfitting in machine learning and how to deal with it*. Last accessed 20 February 2019. 2016. URL: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.

[5] E. Bloedorn, I. Mani e T. R. MacMillan. "Machine learning of user profiles: Representational issues". Em: *arXiv preprint cmp-lg/9712002* (1997).

[6] J. Brownlee. *Supervised and Unsupervised Machine Learning Algorithms*. Last accessed 20 February 2019. 2016. URL: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.

[7] J. Brownlee. *Logistic Regression for Machine Learning*. Last accessed 17 July 2019. 2019. URL: https://machinelearningmastery.com/logistic-regression-for-machine-learning/.

[8] R. D. Burke, K. J. Hammond, V. Kulyukin, S. L. Lytinen, N. Tomuro e S. Schoenberg. "Question answering from frequently asked question files: Experiences with the faq finder system". Em: *AI magazine* 18.2 (1997), p. 57.

[9] K. Chaudhuri. *CSE 151 Machine Learning*. Last accessed 17 July 2019. 2019. URL: https://cseweb.ucsd.edu/classes/sp12/cse151-a/lecture9-final.pdf.

[10] B. C. Chiu e G. I. Webb. "Using decision trees for agent modeling: improving prediction performance". Em: *User Modeling and User-Adapted Interaction* 8.1-2 (1998), pp. 131–152.

[11] M. Dastani, N. Jacobs, C. M. Jonker e J. Treur. "Modeling user preferences and mediating agents in electronic commerce". Em: *Agent Mediated Electronic Commerce*. Springer, 2001, pp. 163–193.

[12] M. Documentation. *Introduction to Azure Blob storage*. Last accessed 2 June 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction.

[13] M. Documentation. *Score Model*. Last accessed 25 July 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/score-model.

[14] M. Documentation. *Train Model*. Last accessed 25 July 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/train-model.

[15] M. A. Documentation. *Two-Class Bayes Point Machine*. Last accessed 17 July 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-bayes-point-machine.

[16] M. A. Documentation. *Two-Class Decision Forest*. Last accessed 17 July 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-decision-forest.

[17] M. A. Documentation. *Two-Class Decision Jungle*. Last accessed 17 July 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-decision-jungle.

[18] R. Documentation. *Decision Tree*. Last accessed 17 July 2019. 2019. URL: https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/parallel_decision_tree.html.

[19] D. R. Fesenmaier, F. Ricci, E. Schaumlechner, K. Wöber, C. Zanella et al. *DIETORECS: Travel advisory for multiple decision styles*. na, 2003.

[20] R. Gandhi. *Bayes Classifier*. Last accessed 18 July 2019. 2019. URL: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c.

[21] P. Giangrandi e C. Tasso. "Managing temporal knowledge in student modeling". Em: *User Modeling*. Springer. 1997, pp. 415–426.

[22] N. Guerra. Private Communication. CreateIT, Lisbon, Portugal, 2019.

[23] M. A. Hall. "Correlation-based feature selection for machine learning". Em: (1999).

[24] S. Ingrez. Private Communication. CreateIT, Lisbon, Portugal, 2019.

[25] R. D.-K. J.Breuker. *Frontiers in Artificial Intelligence and Applications*. Vol. 160. IOS Press.

[26] D. A. Jiménez e C. Lin. "Dynamic branch prediction with perceptrons". Em: *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*. IEEE. 2001, pp. 197–206.

[27] W. Koehrsen. *Overfitting vs. Underfitting: A Conceptual Explanation*. Last accessed 20 February 2019. 2018. URL: https://towardsdatascience.com/overfitting-vs-underfitting-a-conceptual-explanation-d94ee20ca7f9.

[28] S. B. Kotsiantis, I Zaharakis e P Pintelas. "Supervised machine learning: A review of classification techniques". Em: *Emerging artificial intelligence applications in computer engineering* 160 (2007), pp. 3–24.

[29] E. Leopold e J. Kindermann. "Text categorization with support vector machines. How to represent texts in input space?" Em: *Machine Learning* 46.1-3 (2002), pp. 423–444.

[30] Microsoft. *What is the Cloud Solution Provider (CSP) program?* Last accessed 21 January 2019. 2017. URL: https://www.microsoftpartnercommunity.com/t5/Partnership-101/What-is-the-Cloud-Solution-Provider-CSP-program/td-p/2453.

[31] J. B. Morin. *What's the Difference Between a Direct and an Indirect CSP?* Last accessed 20 February 2019. 2017. URL: https://www.sherweb.com/blog/direct-indirect-csp-difference/.

[32] M. D. Mulvenna, S. S. Anand e A. G. Büchner. "Personalization on the Net using Web mining: introduction". Em: *Communications of the ACM* 43.8 (2000), pp. 122–125.

[33] K. P. Murphy et al. "Naive bayes classifiers". Em: *University of British Columbia* 18 (2006), p. 60.

[34] M. Opala. *Amazon AWS, Microsoft Azure, Google Cloud and IBM Watson Comparison: How to Choose the Right Cloud-Based Machine Learning Platform for Your Business?* Last accessed 5 July 2019. 2018. URL: https://www.netguru.com/blog/amazon-aws-microsoft-azure-google-cloud-and-ibm-watson-comparison-how-to-choose-the-right-cloud-based-machine-learning-platform-for-your-business.

[35] B. Pang, L. Lee e S. Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques". Em: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics. 2002, pp. 79–86.

[36] C. Papatheodorou. "Machine learning in user modeling". Em: *Advanced Course on Artificial Intelligence*. Springer. 1999, pp. 286–294.

[37] C. Penaloza. *What is the difference between the Naive Bayes Classifier and the Bayes Point Machine?* Last accessed 18 July 2019. 2019. URL: https://www.quora.com/What-is-the-difference-between-the-Naive-Bayes-Classifier-and-the-Bayes-Point-Machine.

[38] J. R. Quinlan. "Simplifying decision trees". Em: *International journal of man-machine studies* 27.3 (1987), pp. 221–234.

[39] B. Raskutti e A. Beitz. "Acquiring user preferences for information filtering in interactive multi-media services". Em: *Pacific Rim International Conference on Artificial Intelligence*. Springer. 1996, pp. 47–58.

[40] P. Resnick e H. R. Varian. "Recommender systems". Em: *Communications of the ACM* 40.3 (1997), pp. 56–58.

[41] R. Ribeiro. Private Communication. CreateIT, Lisbon, Portugal, 2019.

[42] E. Rich. "Users are individuals: individualizing user models". Em: *International journal of human-computer studies* 51.2 (1999), pp. 323–338.

[43] A. Services. *Amazon AWS, Serviços usados e custos*. Last accessed 5 July 2019. 2019. URL: https://aws.amazon.com/pt/getting-started/projects/build-machine-learning-model/services-costs/.

[44] A. Services. *Guia do desenvolvedor » O que é o Amazon Machine Learning?* Last accessed 5 July 2019. 2019. URL: https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/what-is-amazon-machine-learning.html.

[45] M. Services. *Preços de Machine Learning Studio*. Last accessed 5 July 2019. 2019. URL: https://azure.microsoft.com/pt-pt/pricing/details/machine-learning-studio/.

[46] M. A. Services. *Machine Learning module descriptions*. Last accessed 5 July 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/machine-learning-module-descriptions.

[47] M. A. Services. *Machine Learning module descriptions*. Last accessed 11 July 2019. 2019. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/clean-missing-data.

[48] S. SHARMA. *What the Hell is Perceptron?* Last accessed 20 February 2019. 2017. URL: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53.

[49] B. Shetty. *Supervised Machine Learning: Classification*. Last accessed 11 July 2019. 2018. URL: https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d.

[50] e.-D. Universidade de São Paulo. *Machine learning cábula do algoritmo para o Azure Machine Learning Studio*. Last accessed 12 July 2019. 2018. URL: https://edisciplinas.usp.br/pluginfile.php/3769787/mod_resource/content/1/09_RegressaoLogistica.pdf.

[51] techopedia. *Data Aggregation*. Last accessed 12 Setember 2019. 2019. URL: https://www.techopedia.com/definition/14647/data-aggregation.

[52] L. G. Valiant. "A theory of the learnable". Em: *Communications of the ACM* 27.11 (1984), pp. 1134–1142.

[53] B. C. VATAPÁ. *A granularidade de dados no Data Warehouse*. Last accessed 12 Setember 2019. 2019. URL: http://bicomvatapa.blogspot.com/2014/08/granularidade-granularidade-de-dados-no.html.

[54] G. I. Webb, M. J. Pazzani e D. Billsus. "Machine learning for user modeling". Em: *User modeling and user-adapted interaction* 11.1-2 (2001), pp. 19–29.

[55] G. Weber. "Adaptive learning systems in the World Wide Web". Em: *UM99 User Modeling*. Springer, 1999, pp. 371–377.

[56] G. Widmer e M. Kubat. "Learning in the presence of concept drift and hidden contexts". Em: *Machine learning* 23.1 (1996), pp. 69–101.

[57] Wikipedia. *Perceptron*. Last accessed 11 July 2019. 2019. URL: https://en.wikipedia.org/wiki/Perceptron.

[58] X. Zhang. *Machine learning cábula do algoritmo para o Azure Machine Learning Studio*. Last accessed 12 July 2019. 2018. URL: https://docs.microsoft.com/pt-pt/azure/machine-learning/studio/algorithm-cheat-sheet.

[59] X. Zhu, Z. Ghahramani e J. D. Lafferty. "Semi-supervised learning using gaussian fields and harmonic functions". Em: *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 2003, pp. 912–919.

APÊNDICE A

Lista de métricas iniciais do conjunto de dados para subscrições

Propriedade	Tipo	Descrição
Id	numérico	Identificador interno da entrada da tabela
CustomerId	string	Identificador interno do cliente
SubscriptionId	string	Identificador interno da subscrição
ResellerId	string	Identificador interno do revendedor
ResellerCurrentPriceMargin	numérico	Margem de preço aplicada ao revendedor
CustomerCurrentPriceMargin	numérico	Margem de preço aplicada ao cliente
Date	data	Data da correspondente à entrada da tabela
Quantity	númerico	Quantidade de licenças compradas
TotalLicenseCost	númerico	Custo total acumulado até ao momento (licença)
DailyLicenseCost	númerico	Custo correspondente ao dia em questão (licença)
TotalUsageCost	númerico	Custo total acumulado até ao momento (utilização)
DailyUsageCost	númerico	Custo correspondente ao dia em questão (utilização)
SubscriptionCurrentPriceMargin	númerico	Margem de preço aplicada sobre toda a subscrição
ProviderId	númerico	Identificador interno do fornecedor
TotalLicenseCostForReseller	númerico	Custo total acumulado até ao momento para o revendedor
TotalLicenseCostForCustomer	númerico	Custo total acumulado até ao momento para o cliente
DailyLicenseCostForReseller	númerico	Custo diário associado ao revendedor (licença)
DailyLicenseCostForCustomer	númerico	Custo diário associado ao cliente (licença)
TotalUsageCostForReseller	númerico	Custo total acumulado até ao momento para o revendedor (utilização)
TotalUsageCostForCustomer	númerico	Custo total acumulado até ao momento para o cliente (utilização)
DailyUsageCostForReseller	númerico	Custo diário associado ao revendedor (utilização)
DailyUsageCostForCustomer	númerico	Custo diário associado ao cliente (utilização)
ResellerPriceMarginRule	string	Tipo de regra imposta sobre o preço ao revendedor
CustomerPriceMarginRule	string	Tipo de regra imposta sobre o preço ao cliente
SubscriptionPriceMarginRule	string	Tipo de regra imposta sobre o preço sobre a subscrição

Tabela A.1: Descrição das métricas pertencentes ao conjunto de dados inicial para os dados de subscrições

Apêndice B

Lista de métricas iniciais do conjunto de dados para utilização

Propriedade	Tipo	Descrição
Id	string	Identificador interno da entrada da tabela
PartnerId	string	Identificador interno do parceiro
PartnerName	string	Sigla do parceiro
PartnerBillable AccountId	string	Identificador da conta faturável do parceiro
CustomerCompanyName	string	Nome da empresa do cliente
MpnId	string	Microsoft Partner Network Id
Tier2MpnId	string	Microsoft Partner Network Id de grau 2
InvoiceNumber	string	Código de faturação
ChargeStartDate	data	Data de inicio da faturação
ChargeEndDate	data	Data de fim da faturação
SubscriptionId	string	Identificador interno da subscrição
SubscriptionName	string	Nome da subscrição
SubscriptionDescription	string	Descrição da subscrição
OrderId	string	Identificador interno da encomenda
ServiceName	string	Nome do serviço
ServiceType	string	Tipo de serviço associado
ResourceGuid	string	Identificador do recurso
ResourceName	string	Nome do recurso
Region	string	Região de faturação
Sku	string	Identificador da Stock Keeping Unit, representa um tipo de item especifico
DetailLineItemId	númerico	Identificador dos itens em casos que diferentes unidades são faturadas com diferentes valores
ConsumedQuantity	númerico	Quantidade de unidades consumidas
IncludedQuantity	númerico	Quantidade de unidades incluidas na encomenda
OverageQuantity	númerico	Quantidade consumida acima da utilização permitida
PretaxCharges	númerico	Preço cobrado antes de impostos
TaxAmount	númerico	Quantidade de impostos cobrado
PostTaxTotal	númerico	Preço cobrado após impostos
Currency	string	Moeda na qual é faturada a despesa
PretaxEffectiveRate	númerico	Preço efetivo antes dos impostos
PostTaxEffectiveRate	númerico	Preço efetivo após os impostos
ChargeType	string	Modo de cobrança

Tabela B.1: 1ª Parte: Descrição das métricas pertencentes ao conjunto de dados inicial para os dados de utilização

Propriedade	Tipo	Descrição
DomainName	string	Nome do dominio do cliente
ResellerId	string	Identificador do revendedor
ResellerName	string	Nome do revendedor
ResellerIndirect PartnerInternalId	string	Identificador interno do parceiro indireto do revendedor
CustomerId	string	Identificador interno do cliente
ExternalCustomerId	string	Identificador externo do cliente
ResellerPriceMargin	númerico	Margem de preço aplicado ao revendedor
CustomerPriceMargin	númerico	Margem de preço aplicado ao cliente
SubscriptionPriceMargin	númerico	Margem de preço aplicado sobre a subscrição
InvoiceDate	data	Data de faturação
Unit	string	Unidade de medida para a utilização Azure
ProviderId	string	Identificador do fornecedor
PriceListId	string	Identificador para a lista de preços
ProvidersDataInJsonFormat	string	Informação do fornecedor
ExternalSubscriptionId	string	Identificador externo da subscrição
ResellerPriceMarginRule	string	Regra aplicada ao tipo de preço do revendedor
CustomerPriceMarginRule	string	Regra aplicada ao tipo de preço do cliente
SubscriptionPriceMarginRule	string	Regra aplicada ao tipo de preço da subscrição
ListPriceForReseller	númerico	Preço a cobrar por unidade para o revendedor
ListPriceForCustomer	númerico	Preço a cobrar por unidade para o cliente
PretaxChargesForReseller	númerico	Preço a cobrar antes de impostos ao revendedor
PretaxChargesForCustomer	númerico	Preço a cobrar antes de impostos ao cliente
TaxAmountForReseller	númerico	Valor de imposto a cobrar ao revendedor
TaxAmountForCustomer	númerico	Valor de imposto a cobrar ao cliente
PostTaxTotalForReseller	númerico	Valor total a cobrar ao revendedor já com impostos
PostTaxTotalForCustomer	númerico	Valor total a cobrar ao cliente já com impostos
PretaxEffectiveRateForReseller	númerico	Preço efetivo antes de impostos para revendedor
PretaxEffectiveRateForCustomer	númerico	Preço efetivo antes de impostos para cliente
PostTaxEffectiveRateForReseller	númerico	Preço efetivo após impostos para revendedor
PostTaxEffectiveRateForCustomer	númerico	Preço efetivo após impostos para cliente
ExternalOfferId	string	Identificador externo da oferta
BillingCycleType	string	Periodícidade de faturação

Tabela B.2: 2ª Parte: Descrição das métricas pertencentes ao conjunto de dados inicial para os dados de utilização

Apêndice C

Matriz de desempenhos dos modelos para deteção de situações de churn com dados de subscrições

Algoritmos em Análise

Previsão de Churn	Customer Apenas	Customer+Sub+Date	Date - Sub+Prod	Date - Sub+Prod+Service	ProcDate - Sub+Prod	ProcDate - Sub+Prod+Service
Averaged Perceptron	Perfeito	Excluído	Perfeito	Perfeito	Excluído	Excluído
Logistic Regression	Perfeito	Bom	Perfeito	Perfeito	Excluído	Excluído
Bayes Point Machine	Perfeito	Bom	Perfeito	Perfeito	Excluído	Excluído
Decision Forest	Excluído	Excluído	Excluído	Excluído	Excluído	Excluído
Boosted Decision Tree	Perfeito	Bom	Perfeito	Perfeito	Bom	Bom
Decision Jungle	Excluído	Excluído	Excluído	Excluído	Excluído	Excluído

Agregações de dados em estudo

Tabela C.1: Matriz de desempenhos dos modelos para detecção de situações de churn com dados de subscrições

Apêndice D

Matriz de desempenhos dos modelos para deteção de situações de upgrade com dados de subscrições

Algoritmos em análise

Upgrade Cases	Customer Apenas	Customer+Sub+Date	Date - Sub+Prod	Date - Sub+Prod+Service	ProcDate - Sub+Prod	ProcDate - Sub+Prod+Service
Averaged Perceptron	Bom	Excluído	Excluído	Excluído	Excluído	Excluído
Logistic Regression	Bom+	Excluído	Excluído	Excluído	Excluído	Excluído
Bayes Point Machine	Bom+	Excluído	Excluído	Excluído	Excluído	Excluído
Decision Forest	Bom+	Médio	Médio	Médio	Médio	Médio
Boosted Decision Tree	Quase Perfeito	Médio	Excluído	Médio	Excluído	Excluído
Decision Jungle	Excluído	Médio	Excluído	Médio	Excluído	Excluído

Agregações de dados em estudo

Tabela D.1: Matriz de desempenhos dos modelos para deteção de situações de upgrade com dados de subscrições

Apêndice E

Matriz de desempenhos dos modelos para deteção de situações de downgrade com dados de subscrições

Algoritmos em análise

Downgrade Cases	Agregações de dados em estudo					
	Customer Apenas	**Customer+Sub+Date**	**Date - Sub+Prod**	**Date - Sub+Prod+Service**	**ProcDate - Sub+Prod**	**ProcDate - Sub+Prod+Service**
Averaged Perceptron	Excluído	Excluído	Excluído	Excluído	Excluído	Excluído
Logistic Regression	Excluído	Excluído	Excluído	Excluído	Excluído	Excluído
Bayes Point Machine	Mau	Mau	Excluído	Excluído	Excluído	Excluído
Decision Forest	Excluído	Excluído	Excluído	Excluído	Excluído	Excluído
Boosted Decision Tree	Excluído	Excluído	Excluído	Excluído	Excluído	Excluído
Decision Jungle	Excluído	Excluído	Excluído	Excluído	Excluído	Excluído

Tabela E.1: Matriz de desempenhos dos modelos para deteção de situações de upgrade com dados de subscrições

Apêndice F

Matriz de desempenhos dos modelos para deteção de situações de churn com dados de utilização

Algoritmos em Análise	Agregações de dados em estudo	
Churn Cases	**Usage Racios**	**Usage Taxas**
Averaged Perceptron	Bom	Bom
Logistic Regression	Médio	Médio
Bayes Point Machine	Bom+	Bom+
Decision Forest	Excluído	Excluído
Boosted Decision Tree	Excluído	Excluído
Decision Jungle	Excluído	Excluído

Tabela F.1: Matriz de desempenhos dos modelos para deteção de situações de churn com dados de utilização